
Secretary Ranking with Minimal Inversions

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 We study a secretary problem which captures the task of ranking in online settings.
2 We term this problem the *secretary ranking* problem: elements from an ordered set
3 arrive in random order and instead of picking the maximum element, the algorithm
4 is asked to assign a rank, or position, to each of the elements. The rank assigned
5 is irrevocable and is given knowing only the pairwise comparisons with elements
6 previously arrived. The goal is to minimize the distance of the rank produced to the
7 true rank of the elements measured by the Kendall-Tau distance, which corresponds
8 to the number of pairs that are inverted with respect to the true order.

9 Our main result is a matching upper and lower bound for the secretary ranking
10 problem. We present an algorithm that ranks n elements with only $O(n^{3/2})$ in-
11 versions in expectation, and show that any algorithm necessarily suffers $\Omega(n^{3/2})$
12 inversions when there are n available positions. In terms of techniques, the analysis
13 of our algorithm draws connections to linear probing in the hashing literature, while
14 our lower bound result relies on a general anti-concentration bound for a generic
15 balls and bins sampling process. We also consider the case where the number of
16 positions m can be larger than the number of secretaries n and provide an improved
17 bound by showing a connection of this problem with random binary trees.

18 1 Introduction

19 The secretary problem is one of the first problems studied in online algorithms—in fact, it was
20 extensively studied much before the field of online algorithms even existed. It first appeared in print
21 in 1960 as a recreational problem in Martin Gardner’s Mathematical Games column in Scientific
22 American. In the subsequent decade it caught the attention of many of the eminent probabilist
23 researchers like Lindley [Lin61], Dynkin [Dyn63], Chow et al. [CMRS64] and Gilbert and Mosteller
24 [GM06] among others. In a very entertaining historical survey, Ferguson [Fer89] traces the origin of
25 the secretary problem to much earlier: Cayley in 1875 and Kepler in 1613 pose questions in the same
26 spirit as the secretary problem.

27 Secretary problem has been extended in numerous directions, see for example the surveys by
28 Sakaguchi [Sak95] and Freeman [Fre83]. The problem has had an enormous influence in computer
29 science and has provided some of the basic techniques in the field of online and approximation
30 algorithms. Babaioff et al extended this problem to matroid set systems [BIK07] and Knapsack
31 [BIKK07] and perhaps more importantly, show that the secretary problem is a natural tool for
32 designing online auctions. In the last decade, the secretary problem has also been extended to
33 posets [KLVV11], submodular systems [BHZ10], general set systems [Rub16], stable matchings
34 [BEF⁺17], non-uniform arrivals [KKN15] and applied to optimal data sampling [GD09], design of
35 prophet inequalities [AKW14, EHL17], crowdsourcing systems [SM13], pricing in online settings
36 [CEFJ14], online linear programming [AWY14] and online ad allocation [FHK⁺10].

37 The (admittedly incomplete) list of extensions and applications in the last paragraph serves to
38 showcase that the secretary problem has traditionally been a vehicle for deriving connections between
39 different subfields of computer science and a testbed of new techniques.

40 **Ranking Secretaries.** We consider a natural variant of the secretary problem that captures ranking
41 from pairwise comparisons in online settings. In the *secretary ranking* problem, instead of selecting
42 the maximum element we are asked to *rank* each arriving element. In the process of deriving
43 the optimal algorithm for this problem, we uncover novel connections between ranking and the
44 technique of linear probing, which is one of the earliest techniques in the hashing literature studied
45 by Knuth [Knu63], and also the expected height of random binary trees.

46 In the traditional secretary problem a decision maker is trying to hire a secretary. There is a total
47 order over n secretaries and the goal of the algorithm is to hire the best secretary. The secretaries
48 are assumed to arrive in a random order and the algorithm can only observe the relative rank of
49 each secretary with respect to the previously interviewed ones. Once a secretary is interviewed,
50 the algorithm needs to decide whether to hire the current one or to irrevocably abandon the current
51 candidate and continue interviewing.

52 In our setting, there are m job positions and n secretaries. There is a known total order on positions.
53 Secretaries arrive in random order and, as before, we can only compare a secretary with previously
54 interviewed ones. In our version, all secretaries will be hired and the decision of the algorithm is in
55 which position to hire each secretary. Each position can be occupied by at most one secretary and
56 hiring decisions are irrevocable. Ideally, the algorithm will hire the best secretary in the best position,
57 the second best secretary in the second best position and so on. The loss incurred by the algorithm
58 corresponds to the pairs that are incorrectly ordered, i.e., pairs where a better secretary is hired in a
59 worse position.

60 We give two examples that illustrate scenarios where irrevocable ranking decisions occur online. The
61 first is in the context of task assignments. For concreteness, consider a consulting firm with teams of
62 different skill levels. Projects of different difficulty arrive in an online fashion and when a project
63 arrives, the firm needs to decide which team will execute. Of course, the most difficult projects should
64 go to the most skillful team. The second example is in the context of reward allocation. Consider a
65 university department that would like to assign the best scholarships available to the best students.
66 However, scholarships arrive one at a time and the school needs to decide which student is assigned
67 that scholarship knowing only the relative quality of the scholarships arrived so far.

68 1.1 Our Results and Techniques

69 The perhaps most natural case of the secretary ranking problem is when the numbers of positions
70 and secretaries are the same, i.e. $m = n$, which we call the dense case. The trivial algorithm that
71 assigns a random empty position for each arriving secretary incurs $\Theta(n^2)$ cost, since each pair of
72 elements has probability $1/2$ of being an inversion. On the other hand, $\Omega(n)$ is a trivial lower bound
73 on the cost of any algorithm because nothing is known when the first element arrives. As such, there
74 is a linear gap between the costs of the trivial upper and lower bounds for this secretary ranking
75 problem. Our main result is an asymptotically tight upper and lower bound on the loss incurred by
76 the algorithms for the secretary ranking problem.

77 **Theorem.** *There is an algorithm for the secretary ranking problem that computes a ranking with*
78 *$\mathcal{O}(n^{3/2})$ inversions in expectation. Moreover, any algorithm for this problem makes $\Omega(n^{3/2})$ inver-*
79 *sions in expectation.*

80 There are two challenges in designing an algorithm for secretary ranking. In earlier time steps, there
81 are only a small number of comparisons observed and these do not contain sufficient information
82 to estimate the true rank of the arriving elements. In later time steps, we observe a large number
83 of comparisons and using the randomness of elements arrival, the true rank of the elements can
84 be estimated well. However, the main difficulty is that at these time steps many of the positions
85 have already been assigned to some element arrived earlier and are hence not available. The first
86 information-theoretic challenge exacerbates this second issue. Previous bad placements might imply
87 that all the desired positions are unavailable for the current element, causing a large cost even for an
88 element whose true rank is estimated accurately.

89 The algorithm needs to handle these two opposing challenges simultaneously. The main idea behind
90 our algorithm is to estimate the rank of the current element using the observed comparisons and
91 then add some noise to these estimations to obtain additional randomness in the positions and avoid
92 positively correlated mistakes. We then assign the current element to the closest empty position to
93 this noisy estimated rank. The main technical interest is in the analysis of this algorithm. We draw a
94 connection to the analysis of linear probing in the hashing literature [Knu63] to argue that under this
95 extra noise, there often exists an empty position that is close to the estimated rank.

96 For the lower bound, we analyze the number of random pairwise comparisons needed to estimate
97 the rank of an element accurately. Such results are typically proven in the literature by using anti-
98 concentration inequalities. A main technical difficulty is that most of the existing anti-concentration
99 inequalities are for independent random variables while there is a correlation between the variables
100 we are considering. We prove, to the best of our knowledge, a new anti-concentration inequality for a
101 generic balls in bins problem that involves correlated sampling.

102 In the appendix, we study two additional cases of the secretary ranking problem. In the sparse case,
103 we wish to compute how large the number m of positions needs to be such that we incur no inversions.
104 Clearly for $m = 2^{n+1} - 1$ it is possible to obtain zero inversions with probability 1 and for any
105 number less than that it is also clear that any algorithm needs to cause inversions with non-zero
106 probability. If we only want to achieve zero inversions with high probability, how large does m need
107 to be? By showing a connection between the secretary problem and random binary trees, we show
108 that for $m \geq n^\alpha$ for $\alpha \approx 2.998$ it is possible to design an algorithm that achieves zero inversion
109 with probability $1 - o(1)$. The constant α here is obtained using the high probability bound on the
110 height of a random binary tree of n elements. Finally, we combine the algorithms for the dense and
111 sparse cases to obtain a general algorithm with a bound on the expected number of inversions which
112 smoothly interpolates between the bounds obtained for the dense and sparse cases.

113 1.2 Related Work

114 Our work is inserted in the vast line of literature on the secretary problem, which we briefly discussed
115 earlier. There has been a considerable amount of work on multiple-choice secretary problems
116 where, instead of the single best element, multiple elements can be chosen as they arrive online
117 [Kle05, BIKK07, BIK07, BHZ10, Rub16, KP09]. We note that in multiple-choice secretary problems,
118 the decision at arrival of an element is still binary, whereas in secretary ranking one of n positions
119 must be chosen. More closely related to our work is a paper of Babichenko et al. [BEF⁺17] where
120 elements that arrive must also be assigned to a position. However, the objective is different and the
121 goal, which uses a game-theoretic notion of stable matching, is to maximize the number of elements
122 that are not in a blocking pair. Gobel et al. [GKT15] also studied an online appointment scheduling
123 problem in which the goal is to assign starting dates to a set of jobs arriving online. The objective
124 here is again different from the secretary ranking problem and is to minimize the total weight time of
125 the jobs.

126 Another related line of work in machine learning is the well-known problem of learning to rank that
127 has been extensively studied in recent years (e.g. [BSR⁺05, CQL⁺07, BRL07, XLW⁺08]). Two im-
128 portant applications of this problem are search engines for document retrieval [L⁺09, RJ05, LXQ⁺07,
129 CXL⁺06, XL07] and collaborative filtering approaches to recommender systems [SLH10, SKB⁺12,
130 LY08, WRdVR08]. There has been significant interest recently in ranking from pairwise compar-
131 isons [FRPU94, BFSC⁺13, CS15, SW17, JKSO16, HSRW16, DKMR14, BMW16, AAK17]. To
132 the best of our knowledge, there has not been previous work on ranking from pairwise comparisons
133 in an online setting.

134 Finally, we also briefly discuss hashing, since our main technique is related to linear probing. Linear
135 probing is a classic implementation of hash tables and was first analyzed theoretically by Knuth
136 in 1963 [Knu63], in a report which is now regarded as the birth of algorithm analysis. Since then,
137 different variants of this problem mainly for hash functions with limited independence have been
138 considered in the literature [SS90, PPR07, PT10]. Reviewing the vast literature on this subject is
139 beyond the scope of our paper and we refer the interested reader to these papers for more details.

140 **Organization.** The remainder of the paper is organized as follows. In Section 2 we formalize the
141 secretary ranking problem. In Section 3, we present and analyze our algorithm. Section 4 is devoted
142 to showing the lower bound. Our results for the case where the number of position m is different

143 from the number of elements n appear in Appendix B and Appendix C. Missing proofs and standard
 144 concentration bounds are postponed to the appendix as well.

145 2 Problem Setup

146 In the secretary ranking problem, there are n elements a_1, \dots, a_n that arrive one at a time in an
 147 online manner and in a uniformly random order. There is a total ordering among the elements, but
 148 the algorithm has only access to pairwise comparisons among the elements that have already arrived.
 149 In other words, at time t , the algorithm only observes whether $a_i < a_j$ for all $i, j \leq t$.

150 We define the rank function $\text{rk} : \{a_1, \dots, a_n\} \rightarrow [n]$ as the true rank of the elements in the total
 151 order, i.e., $a_i < a_j$ iff $\text{rk}(a_i) < \text{rk}(a_j)$. Since the elements arrive uniformly at random, $\text{rk}(\cdot)$ is a
 152 random permutation. Upon arrival of an element a_t at time step t , the algorithm must, irrevocably,
 153 place a_t in a position $\pi(a_t) \in [n]$ that is not yet occupied, in the sense that for $a_t \neq a_s$ we must
 154 have $\pi(a_s) \neq \pi(a_t)$. Since the main goal of the algorithm is to place the elements as to reflect the
 155 true rank as close as possible¹, we refer to $\pi(a_t)$ as the *learned rank* of a_t . The goal is to minimize
 156 the number of pairwise mistakes induced by the learned ranking compared to the true ranking. A
 157 pairwise mistake, or an inversion, is defined as a pair of elements a_i, a_j such that $\text{rk}(a_i) < \text{rk}(a_j)$
 158 according to the true underlying ranking but $\pi(a_i) > \pi(a_j)$ according to the learned ranking.

159 The secretary ranking problem generalizes the secretary problem in the following sense: in the
 160 secretary problem, we are only interested in finding the element with the highest rank. However, in
 161 the secretary ranking problem, the goal is to assign a rank to every arrived element and construct a
 162 complete ranking of all elements. Similar to the secretary problem, we make the enabling assumption
 163 that the order of elements arrival is uniformly random.² We measure the cost of the algorithm in
 164 expectation over the randomness of both the arrival order of elements and the algorithm.

165 **Measures of sortedness.** We point out that the primary goal in the secretary ranking problem is
 166 to learn an ordering π of the input elements which is as close as possible to their sorted order. As
 167 such, the *cost* suffered by an algorithm is given by a *measure of sortedness* of π compared to the
 168 true ranking. There are various measures of sortedness studied in the literature depending on the
 169 application. Our choice of using the number of inversions, also known as *Kendall's tau* measure, as
 170 the cost of an algorithm is motivated by the importance of this measure and its close connection to
 171 other measures such as *Spearman's footrule* (see, e.g., Chapter 6B in [Dia88]).

For a mapping $\pi : [n] \rightarrow [n]$, Kendall's tau $K(\pi)$ measures the number of inversions in π , i.e.:

$$K(\pi) := |\{(i, j); (\pi(a_i) - \pi(a_j))(\text{rk}(a_i) - \text{rk}(a_j)) < 0\}|.$$

172 Another important measure of sortedness is Spearman's footrule $F(\pi)$ given by: $F(\pi) :=$
 173 $\sum_{i=1}^n |\text{rk}(a_i) - \pi(a_i)|$, which corresponds to the summation of distances between the true rank
 174 of each element and its current position. A celebrated result of Diaconis and Graham [DG77] shows
 175 that these two measures are within a factor of two of each other, namely, $K(\pi) \leq F(\pi) \leq 2 \cdot K(\pi)$.
 176 We refer to this inequality as the DG inequality throughout the paper. Thus, up to a factor of two, the
 177 goals of minimizing the Kendall tau or Spearman's footrule distances are equivalent and, while the
 178 Kendall tau distance is used in the formulation of the problem, we also use the Spearman's footrule
 179 distance in the analysis.

180 3 The Algorithm

181 In this section, we describe and analyze an algorithm for the secretary ranking problem. Our main
 182 algorithmic result is the following theorem.

183 **Theorem 1.** *There exists an algorithm for the secretary ranking problem that incurs a cost of*
 184 *$O(n\sqrt{n})$ in expectation.*

185 In Section 4, we show that this cost incurred by the algorithm is asymptotically optimal.

¹In other words, hire the better secretaries in better positions.

²It is straightforward to verify that when the ordering is adversarial, any algorithm incurs the trivial cost of $\Omega(n^2)$. For completeness, a proof is provided in Appendix F.

186 **3.1 Description of the Algorithm**

187 The general approach behind the algorithm in Theorem 1 is as follows.

Upon the arrival of element a_t at time step t :

1. **Estimation step:** Estimate the true rank of the arrived element a_t using the *partial* comparisons seen so far.
2. **Assignment step:** Find the nearest currently unassigned rank to this estimate and let $\pi(a_t)$ be this position.

188 We now describe the algorithm in more details. A natural way to estimate the rank of the t -th element
 189 in the estimation step is to compute the rank of this element with respect to the previous $t - 1$ elements
 190 seen so far and then scale this number to obtain an estimate of the rank of this element between 1 and
 191 n . However, for our analysis of the assignment step, we need to tweak this approach slightly: instead
 192 of simply rescaling and rounding, we add perturbation to the estimated rank and then round its value.
 193 This gives a nice distribution of estimated ranks which is crucial for the analysis of the assignment
 194 step. The assignment step then simply assigns a learned rank to the element as close as possible to its
 195 estimated rank. We formalize the algorithm in Algorithm 1.

ALGORITHM 1: Dense Ranking

1 **Input:** a set of n positions, denoted here by $[n]$, and at most n online arrivals.
 2 **for** any time step $t \in [n]$ and element a_t **do**
 3 Define $r_t := |\{a_{t'} \mid a_{t'} < a_t \text{ and } t' < t\}|$.
 4 Sample x_t uniformly in the real interval $[r_t \cdot \frac{n}{t}, (r_t + 1) \cdot \frac{n}{t}]$ and choose $\tilde{rk}(a_t) = \lceil x_t \rceil$.
 5 Set the learned rank of a_t as $\pi(a_t) = \arg \min_{i \in R} |i - \tilde{rk}(a_t)|$ and remove i from R .
 6 **end**

197 We briefly comment on the runtime of the algorithm. By using any self-balancing binary search
 198 tree—such as a red-black tree or an AVL tree—to store the ranking of the arrived elements as well as
 199 the set R of available ranks separately, Algorithm 1 is implementable in $O(\log n)$ time for each step,
 200 so total $O(n \log n)$ worst-case time.

201 We also note some similarity between this algorithm and linear probing in hashing. Linear probing is
 202 an approach to resolving collisions in hashing where, when a key is hashed to a non-empty cell, the
 203 closest neighboring cells are visited until an empty location is found for the key. The similarity is
 204 apparent to our assignment step which finds the nearest currently unassigned rank to the estimated
 205 rank of an element. The analysis of the assignment step follows similar ideas as the analysis for the
 206 linear probing hashing scheme.

207 **3.2 The Analysis**

The total number of inversions can be approximated within a factor of 2 by the Spearman’s footrule.
 Therefore, we can write the cost of Algorithm 1 (up to a factor 2) as follows:

$$\sum_{t=1}^n |\text{rk}(a_t) - \pi(a_t)| \leq \sum_{t=1}^n |\text{rk}(a_t) - \tilde{rk}(a_t)| + \sum_{t=1}^n |\tilde{rk}(a_t) - \pi(a_t)|.$$

208 This basically breaks the cost of the algorithm in two parts: one is the cost incurred by the estimation
 209 step and the other one is the cost of the assignment step. Our analysis then consists of two main parts
 210 where each part bounds one of the terms in the RHS above. In particular, we first prove that given the
 211 partial comparisons seen so far, we can obtain a relatively good estimation to the rank of the arrived
 212 element, and then in the second part, we show that we can typically find an unassigned position in the
 213 close proximity of this estimated rank to assign to it. The following two lemmas capture each part
 214 separately. In both lemmas, the randomness in the expectation is taken over the random arrivals and
 215 the internal randomness of the algorithm:

216 **Lemma 3.1** (Estimation Cost). *In Algorithm 1*, $\mathbb{E} \left[\sum_{t=1}^n \left| \text{rk}(a_t) - \tilde{\text{rk}}(a_t) \right| \right] = O(n\sqrt{n})$.

217 **Lemma 3.2** (Assignment Cost). *In Algorithm 1*, $\mathbb{E} \left[\sum_{t=1}^n \left| \tilde{\text{rk}}(a_t) - \pi(a_t) \right| \right] = O(n\sqrt{n})$.

218 Theorem 1 then follows immediately from these two lemmas and Eq (3.2). The main part of the
 219 argument is the analysis of the assignment cost, i.e., Lemma 3.2, and in particular its connection to
 220 linear probing. The analysis for the estimation cost, i.e., Lemma 3.1, follows from standard Chernoff
 221 bound arguments and is deferred to Appendix D.

Assignment cost: proof of Lemma 3.2. It is useful to think of sampling a random permutation
 in the following recursive way: given a random permutation over $t - 1$ elements, it is possible to
 obtain a random permutation over t elements by inserting the t -th element in a uniformly random
 position between these $t - 1$ elements. Formally, given $\sigma : [t - 1] \rightarrow [t - 1]$, if we sample a position
 i uniformly from $[t]$ and generate permutation $\sigma' : [t] \rightarrow [t]$ such that:

$$\sigma'(t') = \begin{cases} i & \text{if } t' = t \\ \sigma(t') & \text{if } t' < t \text{ and } \sigma'(t') < i \\ \sigma(t') + 1 & \text{if } t' < t \text{ and } \sigma'(t') > i \end{cases}$$

222 then σ' will be a random permutation over t elements. It is simple to see that just by fixing any
 223 permutation and computing the probability of it being generated by this process.

Thinking about sampling the permutation in this way is very convenient for this analysis since at the
 t -th step of the process, the relative order of the first t elements is fixed (even though the true ranks
 can only be determined in the end). In that spirit, let us also define for a permutation $\sigma : [t] \rightarrow [t]$ the
 event \mathcal{O}_σ that σ is the relative ordering of the first t elements:

$$\mathcal{O}_\sigma = \{a_{\sigma(1)} < a_{\sigma(2)} < \dots < a_{\sigma(t)}\}.$$

224 The following proposition asserts that the randomness of the arrival and the inner randomness of the
 225 algorithm, ensures that the estimated ranks at each time step are chosen *uniformly at random* from all
 226 possible ranks in $[n]$.

227 **Proposition 3.3.** *The values of $\tilde{\text{rk}}(a_1), \dots, \tilde{\text{rk}}(a_n)$ are i.i.d and uniformly chosen from $[n]$.*

Proof. First let us show that for any fixed permutation σ over $t - 1$ elements, the relative rank r_t
 defined in the algorithm is uniformly distributed in $\{0, \dots, t - 1\}$. In other words:

$$\Pr[r_t = i \mid \mathcal{O}_\sigma] = \frac{1}{t}, \quad \forall i \in \{0, \dots, t - 1\}.$$

228 Simply observe that there are exactly t permutations over t elements such that the permutation
 229 induced in the first $t - 1$ elements is σ . Since we are sampling a random permutation in this process,
 230 each of these permutation are equally likely to happen. Moreover, since each permutation corresponds
 231 to inserting the t -the element in one of the t positions, we obtain the bound.

232 Furthermore, since the probability of each value of r_t does not depend on the induced permutation
 233 σ over the first $t - 1$ elements, then r_t is independent of σ . Since all the previous values $r_{t'}$ are
 234 completely determined by σ , r_t is independent of all previous $r_{t'}$ for $t' < t$.

235 Finally observe that if r_t is random from $\{0, \dots, t - 1\}$, then x_t is sampled at random from $[0, n]$, so
 236 $\tilde{\text{rk}}(a_t)$ is sampled at random from $[n]$. Since for different values of $t \in [n]$, all r_t are independent, all
 237 the values of $\tilde{\text{rk}}(a_t)$ are also independent. \square

238 Now that we established that $\tilde{\text{rk}}(a_t)$ are independent and uniform, our next task is to bound how
 239 far from the estimated rank we have to go in the assignment step, before we are able to assign a
 240 learned rank to this element. This part of our analysis will be similar to the analysis of the linear
 241 probing hashing scheme. If we are forced to let the learned rank of a_t be far away from $\tilde{\text{rk}}(a_t)$,
 242 say $\left| \tilde{\text{rk}}(a_t) - \pi(a_t) \right| > k$, then this necessarily means that all positions in the integer interval
 243 $[\tilde{\text{rk}}(a_t) - k : \tilde{\text{rk}}(a_t) + k]$ must have already been assigned as a learned rank of some element. In the

244 following, we bound the probability of such an event happening for large values of k compared to the
 245 current time step t .

246 We say that the integer interval $I = [i : i + s - 1]$ of size s is *popular* at time t , iff at least s elements
 247 $a_{t'}$ among the $t - 1$ elements that appear before the t -th element have estimated rank $\tilde{\text{rk}}(a_{t'}) \in I$.
 248 Since by Proposition 3.3 every element has probability s/n of having estimated rank in I and the
 249 estimated ranks are independent, we can bound the probability that I is popular using a standard
 250 application of Chernoff bound (proof deferred to Appendix D).

251 **Claim 3.4.** *Let $\alpha \geq 1$, an interval of size $s \geq 2\alpha \max\left(1, \left(\frac{t}{n-t}\right)^2\right)$ is popular at time t w.p. $e^{-O(\alpha)}$.*

252 We now use the above claim to bound the deviation between $\tilde{\text{rk}}(a_t)$ and $\pi(a_t)$. The following lemma
 253 is the key part of the argument.

254 **Lemma 3.5.** *For any $t \leq n$, we have $\mathbb{E} \left| \tilde{\text{rk}}(a_t) - \pi(a_t) \right| = O\left(\max\left(1, \left(\frac{t}{n-t}\right)^2\right)\right)$.*

255 *Proof.* Fix any $\alpha \geq 1$. We claim that, if the learned rank of a_t is a position which has distance at
 256 least $k_\alpha = 4\alpha \cdot \max\left(1, \left(\frac{t}{n-t}\right)^2\right)$ from its estimated rank, then necessarily there exists an interval I
 257 of length at least $2k_\alpha$ which contains $\tilde{\text{rk}}(a_t)$ and is popular.

258 Let us prove the above claim then. Let I be the shortest integer interval $[a : b]$ which contains $\tilde{\text{rk}}(a_t)$
 259 and moreover both positions a and b are not assigned to a learned rank by time t (by this definition,
 260 $\pi(a_t)$ would be either a or b). For $\left| \tilde{\text{rk}}(a_t) - \pi(a_t) \right|$ to be at least k_α , the length of interval I needs
 261 to be at least $2k_\alpha$. But for I to have length at least $2k_\alpha$, we should have at least $2k$ elements from
 262 a_1, \dots, a_{t-1} to have an estimated rank in I : this is simply because a and b are not yet assigned a
 263 rank by time t and hence any element $a_{t'}$ which has estimated rank outside the interval I is never
 264 assigned a learned rank inside I (otherwise the assignment step should pick a or b , a contradiction).

265 We are now ready to finalize the proof. It is straightforward that in the above argument, it suffices to
 266 only consider the integer intervals $[\tilde{\text{rk}}(a_t) - k_\alpha : \tilde{\text{rk}}(a_t) + k_\alpha]$ parametrized by the choice of $\alpha \geq 1$.
 267 By the above argument and Claim 3.4, for any $\alpha \geq 1$, we have,

$$\begin{aligned} \mathbb{E} \left[\left| \tilde{\text{rk}}(a_t) - \pi(a_t) \right| \right] &\leq \int_{\alpha=0}^{\infty} \Pr \left(\left| \tilde{\text{rk}}(a_t) - \pi(a_t) \right| > k_\alpha \right) \cdot k_\alpha \cdot d\alpha \\ &\leq \int_{\alpha=0}^{\infty} \Pr \left(\text{Integer interval } [\tilde{\text{rk}}(a_t) - k_\alpha : \tilde{\text{rk}}(a_t) + k_\alpha] \text{ is popular} \right) \cdot k_\alpha \cdot d\alpha \\ &\stackrel{\text{Claim 3.4}}{\leq} O\left(\max\left(1, \left(\frac{t}{n-t}\right)^2\right)\right) \cdot \int_{\alpha=0}^{\infty} e^{-O(\alpha)} \cdot \alpha \cdot d\alpha \\ &= O\left(\max\left(1, \left(\frac{t}{n-t}\right)^2\right)\right). \quad \square \end{aligned}$$

268 We are now ready to finalize the proof of Lemma 3.2.

269 *Proof of Lemma 3.2.* We have, $\mathbb{E} \left[\sum_{t=1}^n \left| \tilde{\text{rk}}(a_t) - \pi(a_t) \right| \right] = \sum_{t=1}^n \mathbb{E} \left[\left| \tilde{\text{rk}}(a_t) - \pi(a_t) \right| \right]$ by linear-
 270 ity of expectation. For any $t < n/2$, the maximum term in RHS of Lemma 3.5 is 1 and hence
 271 in this case, we have $\mathbb{E} \left[\left| \tilde{\text{rk}}(a_t) - \pi(a_t) \right| \right] = O(1)$. Thus, the contribution of the first $n/2 - 1$
 272 terms to the above summation is only $O(n)$. Also, when $t > n - \sqrt{n}$, we can simply write
 273 $\mathbb{E} \left[\left| \tilde{\text{rk}}(a_t) - \pi(a_t) \right| \right] \leq n$ which is trivially true and hence the total contribution of these \sqrt{n} sum-
 274 mands is also $O(n\sqrt{n})$. It remains to bound the total contribution of $t \in [n/2, n - \sqrt{n}]$. By
 275 Lemma 3.5, $\sum_{t=n/2}^{n-\sqrt{n}} \mathbb{E} \left[\left| \tilde{\text{rk}}(a_t) - \pi(a_t) \right| \right] \leq O(1) \cdot \sum_{t=n/2}^{n-\sqrt{n}} \left(\frac{t}{n-t}\right)^2 = O(n\sqrt{n})$, where the equal-
 276 ity is by a simple calculation (see Proposition D.3 in Appendix D). \square

277 **4 A Tight Lower Bound**

278 We complement the algorithmic result from the previous section by showing that the cost incurred by
 279 the algorithm is asymptotically optimal.

280 **Theorem 2.** *Any algorithm for the secretary ranking problem incurs $\Omega(n\sqrt{n})$ cost in expectation.*

281 To prove Theorem 2, we first show that no deterministic algorithm can achieve better than $O(n\sqrt{n})$
 282 inversions and then use Yao’s minimax principle to extend the lower bound to randomized algorithms
 283 (by simply fixing the randomness of the algorithm to obtain a deterministic one with the same
 284 performance over the particular distribution of the input).

285 The main ingredient of our proof of Theorem 2 is an anti-concentration bound for sampling without
 286 replacement which we cast as a balls in bins problem. We start by describing this balls in bin problem
 287 and prove the anti-concentration bound in Lemma 4.1. Lemma 4.2 then connects the problem of
 288 online ranking to the balls in bins problem.

289 To continue, we introduce some asymptotic notation that is helpful for readability. We write $v =$
 290 $\Theta_1(n)$ if variable v is linear in n , but also smaller and bounded away from n , i.e., $v = cn$ for some
 291 constant c such that $0 < c < 1$.

292 **Lemma 4.1.** *Assume there are n balls in a bin, r of which are red and the remaining $n - r$ are blue.
 293 Suppose $t < \min(r, n - r)$ balls are drawn from the bin uniformly at random without replacement,
 294 and let $\mathcal{E}_{k,t,r,n}$ be the event that k out of those t balls are red. Then, if $r = \Theta_1(n)$ and $t = \Theta_1(n)$,
 295 for every $k \in \{0, \dots, t\}$: $\Pr(\mathcal{E}_{k,t,r,n}) = O(1/\sqrt{n})$.*

296 Our high level approach toward proving Lemma 4.1 is as follows:

- 297 1. We first use a counting argument to show that $\Pr(\mathcal{E}_{k,t,r,n}) = \binom{r}{k} \binom{n-r}{t-k} / \binom{n}{t}$.
- 298 2. We then use Stirling’s approximation to show $\binom{r}{k} \binom{n-r}{t-k} / \binom{n}{t} = O(n^{-1/2})$ for $k = \lfloor \frac{tr}{n} \rfloor$.
- 299 3. Finally, with a max. likelihood argument, we show that $\arg \max_{k \in [n]} \binom{r}{k} \binom{n-r}{t-k} / \binom{n}{t} \approx \frac{tr}{n}$.

300 By combining these, we have, $\Pr(\mathcal{E}_{k,t,r,n}) \leq \max_{k \in [n]} \binom{r}{k} \binom{n-r}{t-k} / \binom{n}{t} \leq \binom{r}{k^*} \binom{n-r}{t-k^*} / \binom{n}{t}$ for $k^* \approx$
 301 $\frac{tr}{n}$ (by the third step), which we bounded by $O(n^{-1/2})$ (in the second step). The actual proof is
 302 however rather technical and is postponed to Appendix E.

303 The next lemma shows that upon arrival of a_t , any position has probability at least $O(1/\sqrt{n})$ of
 304 being the correct rank for a_t , under some mild conditions. The proof of this lemma uses the previous
 305 anti-concentration bound for sampling without replacement by considering the elements smaller than
 306 a_t to be the red balls and the elements larger than a_t to be the blue balls. For a_t to have rank r and be
 307 the k th element in the ranking so far, the first $t - 1$ elements previously observed must contain $k - 1$
 308 red balls out of the $r - 1$ red balls and $t - k$ blue balls out of the $n - r$ blue balls.

309 **Lemma 4.2.** *Fix any permutation σ of $[t]$ and let \mathcal{O}_σ denote the event that $a_{\sigma(1)} < a_{\sigma(2)} < \dots <$
 310 $a_{\sigma(t)}$. If $\sigma(k) = t$, $k = \Theta_1(t)$ and $t = \Theta_1(n)$ then for any r : $\Pr(\text{rk}(a_t) = r \mid \mathcal{O}_\sigma) = O(1/\sqrt{n})$.*

311 *Proof.* Define \mathcal{E}_k as the event that “ a_t is the k -th smallest element in a_1, \dots, a_t ”. We first have,
 312 $\Pr(\text{rk}(a_t) = r \mid \mathcal{O}_\sigma) = \Pr(\text{rk}(a_t) = r \mid \mathcal{E}_k)$. This is simply because $\text{rk}(a_t)$ is only a function of the
 313 pairwise comparisons of a_t with other elements and does not depend on the ordering of the remaining
 314 elements between themselves. Moreover,

$$\Pr(\text{rk}(a_t) = r \mid \mathcal{E}_k) = \Pr(\mathcal{E}_k \mid \text{rk}(a_t) = r) \cdot \frac{\Pr(\text{rk}(a_t) = r)}{\Pr(\mathcal{E}_k)} = \Pr(\mathcal{E}_k \mid \text{rk}(a_t) = r) \cdot \frac{t}{n}$$

315 since a_t is randomly partitioned across the $[n]$ elements. Notice now that conditioned on $\text{rk}(a_t) = r$,
 316 the event \mathcal{E}_k is exactly the event $\mathcal{E}_{k-1,t-1,r-1,n-1}$ in the sampling without replacement process
 317 defined in Lemma 4.1. The $n - 1$ balls are all the elements but a_t , the $r - 1$ red balls correspond
 318 to elements smaller than a_t , the $n - r$ blue balls to elements larger than a_t , and $t - 1$ balls drawn
 319 are the elements arrived before a_t . Finally, observe that $\Pr(r < k \mid \mathcal{E}_k) = 0$, so for $r < k$, the
 320 bound holds trivially. In the remaining cases, $r = \Theta_1(n)$ and we use the bound in Lemma 4.1 with
 321 $t/n = \Theta(1)$. \square

322 Using the previous lemma, we can lower bound the cost due to the t -th element. Fix any deterministic
323 algorithm \mathcal{A} for the online ranking problem. Recall that $\pi(a_t)$ denotes the learned rank of the item
324 a_t arriving in the t -th time step. For any time step $t \in [n]$, we use $\text{cost}_{\mathcal{A}}(t)$ to denote the cost
325 incurred by the algorithm \mathcal{A} in positioning the item a_t . More formally, if $\text{rk}(a_t) = i$, we have
326 $\text{cost}_{\mathcal{A}}(t) := |i - \pi(a_t)|$. Theorem 2 then follows by Yao's minimax principle principle and the
327 following lemma, whose proof appears in the appendix

328 **Lemma 4.3.** *Fix any deterministic algorithm \mathcal{A} . For any $t = \Theta_1(n)$, $\mathbb{E}[\text{cost}_{\mathcal{A}}(t)] = \Omega(\sqrt{n})$.*

References

- 329
- 330 [AAAK17] Arpit Agarwal, Shivani Agarwal, Sepehr Assadi, and Sanjeev Khanna. Learning with
331 limited rounds of adaptivity: Coin tossing, multi-armed bandits, and ranking from
332 pairwise comparisons. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 39–75, 2017.
- 333
- 334 [AKW14] Pablo D Azar, Robert Kleinberg, and S Matthew Weinberg. Prophet inequalities with
335 limited information. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1358–1377. Society for Industrial and Applied
336 Mathematics, 2014.
- 337
- 338 [AWY14] Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A dynamic near-optimal algorithm for
339 online linear programming. *Operations Research*, 62(4):876–890, 2014.
- 340 [BEF⁺17] Yakov Babichenko, Yuval Emek, Michal Feldman, Boaz Patt-Shamir, Ron Peretz, and
341 Rann Smorodinsky. Stable secretaries. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*,
342 pages 243–244, 2017.
- 343
- 344 [BFSC⁺13] Róbert Busa-Fekete, Balazs Szorenyi, Weiwei Cheng, Paul Weng, and Eyke Hüller-
345 meier. Top-k selection based on adaptive sampling of noisy preferences. In *International Conference on Machine Learning*, pages 1094–1102, 2013.
- 346
- 347 [BHZ10] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Morteza Zadimoghadam. Submodular secretary problem and extensions. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 39–52. Springer, 2010.
- 348
- 349
- 350
- 351 [BIK07] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 434–443. Society for Industrial and Applied Mathematics, 2007.
- 352
- 353
- 354
- 355 [BIKK07] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. A knapsack secretary problem with applications. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, pages 16–28. Springer, 2007.
- 356
- 357
- 358 [BMW16] Mark Braverman, Jieming Mao, and S Matthew Weinberg. Parallel algorithms for select and partition with noisy comparisons. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 851–862. ACM, 2016.
- 359
- 360
- 361 [BRL07] Christopher J Burges, Robert Ragno, and Quoc V Le. Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems*, pages 193–200, 2007.
- 362
- 363
- 364 [BSR⁺05] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.
- 365
- 366
- 367 [CEFJ14] Ilan Reuven Cohen, Alon Eden, Amos Fiat, and Łukasz Jeż. Pricing online decisions: Beyond auctions. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on discrete algorithms*, pages 73–91. SIAM, 2014.
- 368
- 369
- 370 [CMRS64] YS Chow, Sigaiti Moriguti, Herbert Robbins, and SM Samuels. Optimal selection based on relative rank (the “secretary problem”). *Israel Journal of mathematics*, 2(2):81–90, 1964.
- 371
- 372
- 373 [CQL⁺07] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.
- 374
- 375
- 376 [CS15] Yuxin Chen and Changho Suh. Spectral mle: Top-k rank aggregation from pairwise comparisons. In *International Conference on Machine Learning*, pages 371–380, 2015.
- 377

- 378 [CXL⁺06] Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon.
379 Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual*
380 *international ACM SIGIR conference on Research and development in information*
381 *retrieval*, pages 186–193. ACM, 2006.
- 382 [Dev86] Luc Devroye. A note on the height of binary search trees. *Journal of the ACM (JACM)*,
383 33(3):489–498, 1986.
- 384 [DG77] Persi Diaconis and Ronald L Graham. Spearman’s footrule as a measure of disarray.
385 *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 262–268,
386 1977.
- 387 [Dia88] Persi Diaconis. Group representations in probability and statistics. *Lecture Notes-*
388 *Monograph Series*, 11:i–192, 1988.
- 389 [DKMR14] Susan Davidson, Sanjeev Khanna, Tova Milo, and Sudeepa Roy. Top-k and clustering
390 with noisy comparisons. *ACM Transactions on Database Systems (TODS)*, 39(4):35,
391 2014.
- 392 [DP09] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the*
393 *Analysis of Randomized Algorithms*. Cambridge University Press, 2009.
- 394 [Dyn63] Eugene B Dynkin. The optimum choice of the instant for stopping a markov process.
395 *Soviet Mathematics*, 4:627–629, 1963.
- 396 [EHLM17] Hossein Esfandiari, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Morteza Mone-
397 mizadeh. Prophet secretary. *SIAM Journal on Discrete Mathematics*, 31(3):1685–1701,
398 2017.
- 399 [Fer89] Thomas S Ferguson. Who solved the secretary problem? *Statistical science*, pages
400 282–289, 1989.
- 401 [FHK⁺10] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S Mirrokni, and Cliff Stein.
402 Online stochastic packing applied to display ad allocation. In *European Symposium on*
403 *Algorithms*, pages 182–194. Springer, 2010.
- 404 [Fre83] PR Freeman. The secretary problem and its extensions: A review. *International*
405 *Statistical Review/Revue Internationale de Statistique*, pages 189–206, 1983.
- 406 [FRPU94] Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with noisy
407 information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994.
- 408 [GD09] Yogesh Girdhar and Gregory Dudek. Optimal online data sampling or how to hire the
409 best secretaries. In *2009 Canadian Conference on Computer and Robot Vision*, pages
410 292–298. IEEE, 2009.
- 411 [GKT15] Oliver Göbel, Thomas Kesselheim, and Andreas Tönnis. Online appointment schedul-
412 ing in the random order model. In *Algorithms - ESA 2015 - 23rd Annual European*
413 *Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 680–692,
414 2015.
- 415 [GM06] John P Gilbert and Frederick Mosteller. Recognizing the maximum of a sequence. In
416 *Selected Papers of Frederick Mosteller*, pages 355–398. Springer, 2006.
- 417 [HSRW16] Reinhard Heckel, Nihar B Shah, Kannan Ramchandran, and Martin J Wainwright.
418 Active ranking from pairwise comparisons and when parametric assumptions don’t
419 help. *arXiv preprint arXiv:1606.08842*, 2016.
- 420 [JKSO16] Minje Jang, Sunghyun Kim, Changho Suh, and Sewoong Oh. Top-*k* ranking from pair-
421 wise comparisons: When spectral ranking is optimal. *arXiv preprint arXiv:1603.04153*,
422 2016.
- 423 [KKN15] Thomas Kesselheim, Robert Kleinberg, and Rad Niazadeh. Secretary problems with
424 non-uniform arrival order. In *Proceedings of the forty-seventh annual ACM symposium*
425 *on Theory of computing*, pages 879–888. ACM, 2015.

- 426 [Kle05] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online
427 auctions. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete*
428 *algorithms*, pages 630–631. Society for Industrial and Applied Mathematics, 2005.
- 429 [KLVV11] Ravi Kumar, Silvio Lattanzi, Sergei Vassilvitskii, and Andrea Vattani. Hiring a secretary
430 from a poset. In *Proceedings of the 12th ACM conference on Electronic commerce*,
431 pages 39–48. ACM, 2011.
- 432 [Knu63] Don Knuth. Notes on “open” addressing. *Citeseer*, 1963.
- 433 [KP09] Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hyper-
434 graphs. In *Automata, Languages and Programming, 36th Internatilonal Colloquium,*
435 *ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II*, pages 508–520,
436 2009.
- 437 [L⁺09] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends®*
438 *in Information Retrieval*, 3(3):225–331, 2009.
- 439 [Lin61] Denis V Lindley. Dynamic programming and decision theory. *Applied Statistics*,
440 10(1):39–51, 1961.
- 441 [LXQ⁺07] Tie-Yan Liu, Jun Xu, Tao Qin, Wenying Xiong, and Hang Li. Letor: Benchmark
442 dataset for research on learning to rank for information retrieval. In *Proceedings of*
443 *SIGIR 2007 workshop on learning to rank for information retrieval*, volume 310. ACM
444 Amsterdam, The Netherlands, 2007.
- 445 [LY08] Nathan N Liu and Qiang Yang. Eigenrank: a ranking-oriented approach to collaborative
446 filtering. In *Proceedings of the 31st annual international ACM SIGIR conference on*
447 *Research and development in information retrieval*, pages 83–90. ACM, 2008.
- 448 [PPR07] Anna Pagh, Rasmus Pagh, and Milan Ruzic. Linear probing with constant inde-
449 pendence. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of*
450 *computing*, pages 318–327. ACM, 2007.
- 451 [PS97] Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring
452 via an extension of the chernoff-hoeffding bounds. *SIAM Journal on Computing*,
453 26(2):350–368, 1997.
- 454 [PT10] Mihai Patrăşcu and Mikkel Thorup. On the k-independence required by linear probing
455 and minwise independence. In *International Colloquium on Automata, Languages,*
456 *and Programming*, pages 715–726. Springer, 2010.
- 457 [Ree03] Bruce Reed. The height of a random binary search tree. *Journal of the ACM (JACM)*,
458 50(3):306–332, 2003.
- 459 [RJ05] Filip Radlinski and Thorsten Joachims. Query chains: learning to rank from implicit
460 feedback. In *Proceedings of the eleventh ACM SIGKDD international conference on*
461 *Knowledge discovery in data mining*, pages 239–248. ACM, 2005.
- 462 [Rub16] Aviad Rubinfeld. Beyond matroids: Secretary problem and prophet inequality with
463 general constraints. In *Proceedings of the forty-eighth annual ACM symposium on*
464 *Theory of Computing*, pages 324–332. ACM, 2016.
- 465 [Sak95] Minoru Sakaguchi. Optimal stopping games: A review. *Mathematica japonicae*,
466 42(2):343–351, 1995.
- 467 [Ser74] Robert J Serfling. Probability inequalities for the sum in sampling without replacement.
468 *The Annals of Statistics*, pages 39–48, 1974.
- 469 [SKB⁺12] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and
470 Alan Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-
471 more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*,
472 pages 139–146. ACM, 2012.

- 473 [SLH10] Yue Shi, Martha Larson, and Alan Hanjalic. List-wise learning to rank with matrix
474 factorization for collaborative filtering. In *Proceedings of the fourth ACM conference*
475 *on Recommender systems*, pages 269–272. ACM, 2010.
- 476 [SM13] Yaron Singer and Manas Mittal. Pricing mechanisms for crowdsourcing markets. In
477 *Proceedings of the 22nd international conference on World Wide Web*, pages 1157–
478 1166. ACM, 2013.
- 479 [SS90] Jeanette P Schmidt and Alan Siegel. The analysis of closed hashing under limited
480 randomness. In *Proceedings of the twenty-second annual ACM symposium on Theory*
481 *of computing*, pages 224–234. ACM, 1990.
- 482 [SW17] Nihar B. Shah and Martin J. Wainwright. Simple, robust and optimal ranking from
483 pairwise comparisons. *Journal of Machine Learning Research*, 18:199:1–199:38, 2017.
- 484 [WRdVR08] Jun Wang, Stephen Robertson, Arjen P de Vries, and Marcel JT Reinders. Probabilistic
485 relevance ranking for collaborative filtering. *Information Retrieval*, 11(6):477–497,
486 2008.
- 487 [XL07] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In
488 *Proceedings of the 30th annual international ACM SIGIR conference on Research and*
489 *development in information retrieval*, pages 391–398. ACM, 2007.
- 490 [XLW⁺08] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach
491 to learning to rank: theory and algorithm. In *Proceedings of the 25th international*
492 *conference on Machine learning*, pages 1192–1199. ACM, 2008.

493 **A Useful Concentration of Measure Inequalities**

494 We use the following two standard versions of Chernoff bound (see, e.g., [DP09]) throughout.

495 **Proposition A.1** (Multiplicative Chernoff bound). *Let X_1, \dots, X_n be n independent random variables taking values in $[0, 1]$ and let $X := \sum_{i=1}^n X_i$. Then, for any $\varepsilon \in (0, 1]$,*

$$\Pr(X \geq (1 + \varepsilon) \cdot \mathbb{E}[X]) \leq \exp(-2\varepsilon^2 \cdot \mathbb{E}[X]).$$

497 **Proposition A.2** (Additive Chernoff bound). *Let X_1, \dots, X_n be n independent random variables taking values in $[0, 1]$ and let $X := \sum_{i=1}^n X_i$. Then,*

$$\Pr(|X - \mathbb{E}[X]| > t) \leq 2 \cdot \exp\left(-\frac{2t^2}{n}\right).$$

499 *Moreover, if X_1, \dots, X_n are negatively correlated (i.e. $\Pr[X_i = 1, \forall i \in S] \leq \prod_{i \in S} \Pr[X_i = 1]$ for all $S \subseteq [n]$), then the upper tail holds: $\Pr(X - \mathbb{E}[X] > t) \leq \exp\left(-\frac{2t^2}{n}\right)$.*

501 Moreover, in the above setting, if X comes from a sampling *with* replacement process, then the inequality holds for both upper and lower tails. For sampling without replacement, we refer to Serfling [Ser74] for a complete discussion and for Chernoff bounds for negatively correlated random variables see [PS97].

505 **Proposition A.3** (Chernoff bound for sampling without replacement). *Consider an urn with $a \geq b$ red and blue balls. Draw b balls uniformly from the urn without replacement and let X be the number of red balls drawn, then the two sided bound holds: $\Pr(|X - \mathbb{E}[X]| > t) \leq 2 \cdot \exp\left(-\frac{2t^2}{b}\right)$.*

508 *Proof.* If X_i is the event that the i -th ball is red, then since X_i are negatively correlated, the upper tail Chernoff bound of $X = \sum_i X_i$ holds. Now, let $Y_i = 1 - X_i$ be the probability that the i -th ball is blue and $Y = \sum_i Y_i$. The upper tail for Y correspond to the lower tail for X , i.e.:
 511 $\Pr(X - \mathbb{E}[X] < t) = \Pr(Y - \mathbb{E}[Y] > t) \leq \exp\left(-\frac{2t^2}{b}\right)$. □

512 **B Sparse Secretary Ranking**

513 In this section, we consider the special case where the number of positions is very large, which we call sparse secretary ranking. In the extreme when $m \geq 2^{n+1} - 1$ it is possible to assign a position to each secretary without ever incurring a mistake. To do that, build a complete binary tree of height n and associate each position in $[m]$ with a node (both internal and leaf) of the binary tree such that the order of the positions corresponds to the pre-order induced by the binary tree (see figure B). Once the elements arrive in an online fashion, insert them in the binary tree and allocate them in the corresponding position.

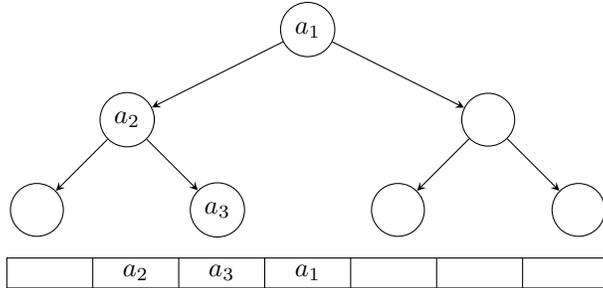


Figure 1: Illustration of the binary tree algorithm for $m = 7$ and order $a_2 < a_3 < a_1$.

520 We note that the above algorithm works for any order of arrival. If the elements arrive in random order, it is possible to obtain zero inversions with high probability for an exponentially smaller value of m . The idea is very similar to the one outlined above. Let H_n be a random variable corresponding to the height of a binary tree built from n elements in random order. Reed [Ree03] shows that

524 $\mathbb{E}[H_n] = \alpha \ln(n)$, $\text{Var}[H_n] = O(1)$ where α is the solution of the equation $\alpha \ln(2e/\alpha) = 1$ which is
 525 $\alpha \approx 4.31107$.

526 Since the arrival order of secretaries is uniformly random, the binary tree algorithm won't touch any
 527 node with height more than $\bar{h} = \lceil (\alpha + O(\epsilon)) \ln(n) \rceil$ with probability $1 - o(1)$. This observation
 528 allows us to define an algorithm that obtains zero inversions with probability $1 - o(1)$. If $m \geq$
 529 $2^{\bar{h}+1} - 1 = \Omega(n^{2.998+\epsilon})$, we can build a binary tree with height \bar{h} and associate each node of the
 530 tree to a position. Once the elements arrive, allocate the item in the corresponding position. If an
 531 item is added to the tree with height larger than \bar{h} , start allocating the items arbitrarily.

532 **Theorem 3.** *If $m \geq n^{2.998+\epsilon}$ then the algorithm that allocates according to a binary tree incurs zero*
 533 *inversions with probability $1 - o(1)$.*

Devroye [Dev86] bounds the tail of the distribution of H_n as follows:

$$\Pr[H_n \geq k \cdot \ln n] \leq \frac{1}{n} \cdot \left(\frac{2e}{k}\right)^{k \cdot \ln n}$$

534 for $k > 2$. In particular: $\Pr[H_n \geq 6.3619 \cdot \ln n] \leq 1/n^2$. Adapting the analysis above, we can show
 535 that for $m \geq 4.41$ (where $4.41 = 6.3619 \cdot \ln(2)$) the algorithm incurs less than one inversion in
 536 expectation.

537 **Corollary 4.** *If $m \geq \Omega(n^{4.41})$ then the algorithm that allocates according to a binary tree incurs*
 538 *$O(1)$ inversion in expectation.*

539 C General Secretary Ranking

540 In the general case, we combine the ideas for the sparse and dense case to obtain an algorithm
 541 interpolating both cases. As described in Algorithm 2, we construct a complete binary search tree of
 542 height h and associating one position for each internal node, but for the leaves we associate a block
 543 of $w = m/2^h - 1$ positions (see Figure 2). If we insert an element in a leaf, we allocate according
 544 to an instance of the dense ranking algorithm. By that we mean that the algorithm pretends that
 545 the elements allocated to that leaf are an isolated instance of dense ranking with w elements and
 546 w positions. We will set h such that in expectation there only w elements in each leaf with high
 547 probability. If at some point more than w elements are placed in any given leaf, the algorithm starts
 548 allocating arbitrarily.

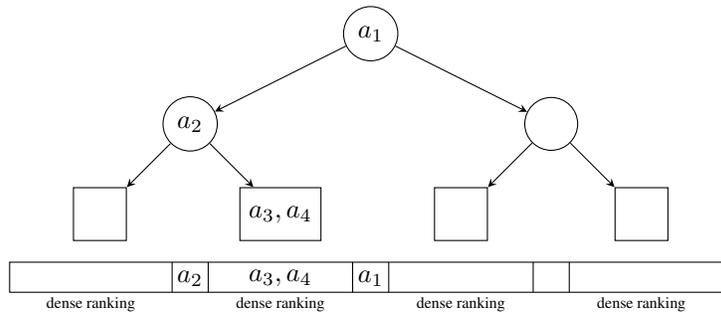


Figure 2: Illustration of the general algorithm (Algorithm 2) for order $a_2 < a_3 < a_4 < a_1$. The leaves are associated with blocks of w consecutive positions and internal nodes are associated with a single position. Elements a_3 and a_4 are associated with the same leaf and therefore we place them in a block of w positions as we would in a dense ranking problem with w arrivals and w positions.

ALGORITHM 2: General secretary ranking

1 **Input:** a set of m positions, at most n online arrivals and a height h .
2 Construct a complete binary search tree T of height h and associate one position for each internal node, and a block of $w = m/2^h - 1$ positions for each leaf such that the order of the positions corresponds to the pre-order induced by the binary tree
3 **for** any time step $t \in [n]$ and element a_t **do**
4 Insert a_t in the tree T
5 **if** a_t reaches an empty internal node **then**
6 Place a_t in the position corresponding to this internal node
7 **end**
8 **else**
9 Place a_t according to an instance of the dense ranking algorithm (Algorithm 1) over the block of positions corresponding to the leaf reached by a_t . If there are no position available in that block, place a_t arbitrarily
10 **end**
11 **end**

For stating our main theorem and its proof, it is convenient to define the functions:

$$f(\alpha) = \frac{\alpha \ln(2) - 1}{1 - 2\alpha \ln(2e/\alpha)} \quad g(\alpha) = \frac{1}{1 - 2\alpha \ln(2e/\alpha)}$$

550 defined in the interval (α_0, ∞) where $\alpha_0 \approx 4.910$ is the solution to the equation $1 - 2\alpha_0 \ln(2e/\alpha_0) =$
551 0 . Both functions are monotone decreasing from $+\infty$ (when $\alpha = \alpha_0$) to zero (when $\alpha \rightarrow \infty$). We
552 are now ready to state our main theorem:

553 **Theorem 5.** Assume $m \geq 10n \log n$ and let $\alpha \in (\alpha_0, \infty)$ be the solution to $\frac{m}{9n \log n} = n^{f(\alpha)}$, then
554 the expected number of inversions of the general secretary ranking algorithm with $h = \alpha \ln(n^{g(\alpha)})$
555 is $\tilde{O}(n^{1.5-0.5g(\alpha)})$.

556 We note that the algorithm smoothly interpolated between the two cases previously analyzed. When
557 $m = n \log(n)$ then $\alpha \rightarrow \infty$, so $g(\alpha) \rightarrow 0$ and the bound on the theorem becomes $\tilde{O}(n^{1.5})$. In the
558 other extreme, when $m \rightarrow \infty$, then $\alpha \rightarrow \alpha_0$ and therefore $g(\alpha) \rightarrow \infty$, so the bound on the number
559 of inversions becomes $O(1)$.

Proof. Let H_t be the height of the binary tree formed by the first t elements. By Devroye's bound [Dev86], the probability that a random binary tree formed by the first $t := n^{g(\alpha)}$ elements has height more than $h = \alpha \ln(t)$ is

$$\Pr[H_t \geq h] \leq \frac{1}{t} (2e/\alpha)^{\alpha \ln t} = t^{\alpha \ln(2e/\alpha) - 1}.$$

In case this event happens, we will use the trivial bound of $O(n^2)$ on the number of inversions, which will contribute

$$n^2 t^{\alpha \ln(2e/\alpha) - 1} = n^{1.5 - 0.5/(1 - 2\alpha \ln(2e/\alpha))} = n^{1.5 - 0.5g(\alpha)}$$

560 to the expectation. From this point on, we consider the remaining event that $H_t < h$.

561 Next, we condition on the first t elements that we denote b_1, \dots, b_t such that $b_1 < \dots < b_t$. We note
562 that for each remaining element $a_i, i > t$, we have $b_j < a_i < b_{j+1}$ with probability $1/(t+1)$ for
563 all $j \in [t]$. Since b_1, \dots, b_t are all placed in positions corresponding to internal nodes, each element
564 has at most probability $1/t$ of hitting any of the dense-ranking instances. Thus, each dense ranking
565 instance receives at most n/t elements in expectation, and by a standard application of the Chernoff
566 bound, the probability that a dense ranking instance sees more than $9(n/t) \log n$ elements is n^{-3} . If
567 this is the case for some dense ranking instance, we again use the n^2 trivial bound, which contributes
568 at most 1 to the expected number of inversions. For the remainder of the proof, we assume that each
569 dense ranking instance gets at most $9(n/t) \log n$ elements.

Next, note that the size of each block is

$$w = \frac{m}{2^h} - 1 = \frac{m}{t^{\alpha \ln(2)}} - 1 \geq 9(n/t) \log n$$

570 where the last equality is by definition of t . Thus, no more than w elements are inserted in any leaf.

Let v_i is the number of elements in each of the dense rank instances. We note that within the elements in each dense ranking block the arrival order is random, so we can apply the bound from Section 3 and obtain by Theorem 1 that the total expected cost from the inversions caused by dense rank is at most

$$\sum_i O(v_i^{1.5}) \leq \tilde{O}(t \cdot (n/t)^{1.5}) = \tilde{O}(n^2 t^{\alpha \ln(2e/\alpha) - 1}) = \tilde{O}(n^{1.5 - 0.5g(\alpha)})$$

571 since $\sum_i v_i = n$ and $v_i \leq (n/t) \log(n)$. By the construction there are no inversions between
 572 elements inserted in different leaves and between an element inserted in an internal node and any
 573 other element. Summing the expected number of mistakes from the events $H_t \geq h$ and $H_t < h$, we
 574 get the bound in the statement of the theorem. \square

575 D Missing Analysis from Section 3

576 **Estimation Cost: Proof of Lemma 3.1.** We begin with the following useful proposition.

577 **Proposition D.1.** *If $1 < t \leq n$ and $0 \leq r \leq t - 1$, then $r \cdot \binom{n}{t} \leq r \cdot \binom{n-1}{t-1} \leq (r+1) \cdot \binom{n}{t}$.*

578 *Proof.* $0 \leq r \left(\frac{n-1}{t-1} - \frac{n}{t} \right) = r \frac{n-t}{t(t-1)} \leq (t-1) \frac{n-t}{t(t-1)} \leq \frac{n}{t}$. \square

579 The correctness of the estimation step in our algorithm relies on the following proposition that
 580 bounds the probability of the deviation between the estimated rank and the true rank of each element
 581 (depending on the time step it arrives). The proof uses the Chernoff bound for sampling without
 582 replacement.

583 **Proposition D.2.** *For any $t > 1$ and any $\alpha \geq 0$, $\Pr \left(\left| \text{rk}(a_t) - \tilde{\text{rk}}(a_t) \right| \geq 1 + \frac{n}{t} + \alpha \cdot \frac{n-1}{\sqrt{t-1}} \right) \leq$
 584 $e^{-\Omega(\alpha^2)}$.*

585 *Proof.* Fix any $t \in [n]$ and element a_t and recall that $\text{rk}(a_t)$ denotes the true rank of a_t . Conditioned
 586 on a fixed value for the rank of a_t , the distribution of the number of elements r_t that arrived before
 587 a_t and have a smaller rank is equivalent to a sampling without replacement process of $t - 1$ balls
 588 where the urn has $\text{rk}(a_t) - 1$ red balls and $n - \text{rk}(a_t)$ blue balls (and the goal is to count the number
 589 of red balls). As such $\mathbb{E}[r_t] = \frac{\text{rk}(a_t) - 1}{n-1}$ and by the Chernoff bound for sampling without replacement
 590 (Proposition A.3 with $a = n$ and $b = t - 1$), we have:

$$\Pr \left(|r_t - \mathbb{E}[r_t]| \geq \alpha \sqrt{t-1} \right) \leq 2 \cdot \exp \left(- \frac{2(\alpha \sqrt{t-1})^2}{t-1} \right) = e^{-\Omega(\alpha^2)}.$$

591 We now argue that

$$\Pr \left(\left| \text{rk}(a_t) - \tilde{\text{rk}}(a_t) \right| \geq 1 + \frac{n}{t} + \alpha \cdot \frac{n-1}{\sqrt{t-1}} \right) \leq \Pr \left(|r_t - \mathbb{E}[r_t]| \geq \alpha \sqrt{t-1} \right),$$

592 which finalizes the proof by the bound in above equation.

593 To see this, note that,

$$\alpha \frac{n-1}{\sqrt{t-1}} \geq \left| \frac{n-1}{t-1} r_t - \text{rk}(a_t) \right| \geq |x_t - \text{rk}(a_t)| - \frac{n}{t} \geq \left| \tilde{\text{rk}}(a_t) - \text{rk}(a_t) \right| - 1 - \frac{n}{t}$$

594 The first inequality follows from substituting the expectation in $|r_t - \mathbb{E}[r_t]| \geq \alpha \sqrt{t-1}$ and multi-
 595 plying the whole expression by $(n-1)/(t-1)$. The second inequality just follows from the fact that
 596 both the variable x_t (defined in step 4 of Algorithm 2) and $\frac{n-1}{t-1} r_t$ are in the interval $[\frac{n}{t} r_t, \frac{n}{t} (r_t + 1)]$.
 597 The fact that x_t is in this interval comes directly from its definition in the algorithm and the fact that
 598 $\frac{n-1}{t-1} r_t$ is in the interval is by a simple calculation (see Proposition D.1 in Appendix D). The last
 599 inequality follows from the fact that $\tilde{\text{rk}}(a_t) = \lceil x_t \rceil$. \square

600 We are now ready to prove Lemma 3.1.

601 *Proof of Lemma 3.1.* Fix any $t > 1$; we have,

$$\begin{aligned} \mathbb{E} \left[\left| \text{rk}(a_t) - \tilde{\text{rk}}(a_t) \right| - 1 - \frac{n}{t} \right] &\leq \int_{\alpha=0}^{\infty} \Pr \left(\left| \text{rk}(a_t) - \tilde{\text{rk}}(a_t) \right| - 1 - \frac{n}{t} \geq \alpha \cdot \frac{n-1}{\sqrt{t-1}} \right) \cdot \frac{n-1}{\sqrt{t-1}} \cdot d\alpha \\ &\leq \frac{n-1}{\sqrt{t-1}} \cdot \int_{\alpha=0}^{\infty} e^{-\Omega(\alpha^2)} \cdot d\alpha = O\left(\frac{n}{\sqrt{t}}\right). \quad (\text{by Proposition D.2}) \end{aligned}$$

602 Hence, using the trivial bound for $t = 1$ and the bound above for $t > 1$ we conclude that:

$$\mathbb{E} \left[\sum_{t=1}^n \left| \text{rk}(a_t) - \tilde{\text{rk}}(a_t) \right| \right] = \sum_{t=1}^n \mathbb{E} \left[\left| \text{rk}(a_t) - \tilde{\text{rk}}(a_t) \right| \right] = \sum_{t=1}^n O\left(\frac{n}{t} + \frac{n}{\sqrt{t}}\right) = O(n\sqrt{n})$$

603 **Missing analysis for Lemma 3.2.**

604 **Claim 3.4.** Let $\alpha \geq 1$, an interval of size $s \geq 2\alpha \max\left(1, \left(\frac{t}{n-t}\right)^2\right)$ is popular at time t w.p. $e^{-O(\alpha)}$.

605 *Proof.* The proof follows directly from the Chernoff bound in Proposition A.1. For $t' \in [t]$, let $X_{t'}$
606 be the event that $\tilde{\text{rk}}(a_{t'}) \in I$ and $X = \sum_{t'=1}^t X_{t'}$, then setting $\epsilon = \min(1, \frac{n-t}{t})$ we have that:

$$\begin{aligned} \Pr(I \text{ is popular}) &= \Pr(X \geq s) \leq \Pr(X > (1 + \epsilon) \cdot \epsilon \cdot \mathbb{E}[X]) \\ &\leq \exp\left(-\frac{\epsilon^2 \cdot \mathbb{E}[X]}{2}\right) = e^{-O(\alpha)} \end{aligned}$$

607 as $\mathbb{E}[X] = s \cdot t/n$. □

608 **Proposition D.3.** For any integer $n > 0$, $\sum_{t=1}^{n-\sqrt{n}} \left(\frac{t}{n-t}\right)^2 = O(n\sqrt{n})$.

609 *Proof.* By defining $k = n - t$, we have,

$$\sum_{t=1}^{n-\sqrt{n}} \left(\frac{t}{n-t}\right)^2 = \sum_{k=\sqrt{n}}^{n-1} \left(\frac{n-k}{k}\right)^2 \leq \sum_{k=\sqrt{n}}^{n-1} \left(\frac{n}{k}\right)^2$$

610 For $i \in [\sqrt{n}]$, define $K_i := \{k \mid i \cdot \sqrt{n} \leq k < (i+1) \cdot \sqrt{n}\}$. For any $k \in K_i$, we have, $\frac{n}{k} \leq \frac{\sqrt{n}}{i}$.
611 As such, we can write,

$$\begin{aligned} \sum_{k=\sqrt{n}}^{n-1} \left(\frac{n}{k}\right)^2 &= \sum_{i=1}^{\sqrt{n}} \sum_{k \in K_i} \left(\frac{n}{k}\right)^2 \leq \sum_{i=1}^{\sqrt{n}} \sum_{k \in K_i} \left(\frac{\sqrt{n}}{i}\right)^2 \\ &\leq \sum_{i=1}^{\sqrt{n}} n \cdot |K_i| \cdot \frac{1}{i^2} \leq n\sqrt{n} \cdot \sum_{i=1}^{\sqrt{n}} \frac{1}{i^2} = O(n\sqrt{n}) \end{aligned}$$

612 as the series $\sum_i \frac{1}{i^2}$ is a converging series. □

613 E Anti-Concentration for Sampling Without Replacement

614 We prove Lemma 4.1 restated here for convenience.

615 **Lemma** (Restatement of Lemma 4.1). Assume there are n balls in a bin, r of which are red and the
616 remaining $n - r$ are blue. Suppose $t < \min(r, n - r)$ balls are drawn from the bin uniformly at
617 random without replacement, and let $\mathcal{E}_{k,t,r,n}$ be the event that k out of those t balls are red. Then, if
618 $r = \Theta_1(n)$ and $t = \Theta_1(n)$, for every $k \in \{0, \dots, t\}$: $\Pr(\mathcal{E}_{k,t,r,n}) = O(1/\sqrt{n})$.

619 To prove Lemma 4.1, we will describe the sampling without replacement process explicitly and
620 bound the relevant probabilities.

Proposition E.1. Let $0 < c < 1$ be a constant. Then:

$$\binom{n}{cn} = \Theta(n^{-1/2} c^{-(cn+1/2)} (1-c)^{-((1-c)n+1/2)})$$

621 The notation $y = \Theta(x)$ in the lemma statement means that there are universal constants $0 < \underline{\alpha} < \bar{\alpha}$
622 independent of c and n such that $\underline{\alpha} \cdot x \leq y \leq \bar{\alpha} \cdot x$. The proof is based on the following version of
623 Stirling's approximation: $\sqrt{2\pi} n^{n+\frac{1}{2}} e^{-n} \leq n! \leq e n^{n+\frac{1}{2}} e^{-n}$. which can be written in our notation
624 as: $n! = \Theta(n^{n+\frac{1}{2}} e^{-n})$. The proof of the previous lemmas follows from just expanding the factorials
625 in the definition of the binomial:

626 *Proof.* Observe that

$$\binom{n}{cn} = \frac{n!}{(cn)!((1-c)n)!} = \Theta\left(\frac{n^{n+\frac{1}{2}} e^{-n}}{(cn)^{cn+\frac{1}{2}} e^{-cn} ((1-c)n)^{(1-c)n+\frac{1}{2}} e^{-((1-c)n)}}\right)$$

627 The statement follows from simplifying the right hand side. \square

Lemma E.2. Assume that $r = \Theta_1(n)$ and $t = \Theta_1(n)$ and $t \leq \min(r, n-r)$, then for $k = \lfloor rt/n \rfloor$, we have

$$\frac{\binom{r}{k} \cdot \binom{n-r}{t-k}}{\binom{n}{t}} = \mathcal{O}(1/\sqrt{n}).$$

Proof. Start by writing $r = c_r \cdot n$ and $t = c_t \cdot n$ for $0 < c_r, c_t < 1$. It will be convenient to assume that $k = rt/n$ is an integer (if not and we need to apply floors, the exact same proof work by keeping track of the errors introduced by floor). Then we can write: First, note that

$$\frac{\binom{r}{k} \cdot \binom{n-r}{t-k}}{\binom{n}{t}} = \frac{\binom{c_r n}{c_t c_r n} \cdot \binom{(1-c_r)n}{(1-c_r)c_t n}}{\binom{n}{c_t n}}$$

We can now apply the approximation in Proposition E.1 obtaining:

$$\Theta\left(\frac{n^{1/2} c_t^{c_t n + \frac{1}{2}} (1-c_t)^{(1-c_t)n + \frac{1}{2}}}{(c_r n)^{1/2} c_t^{c_t(c_r n) + \frac{1}{2}} (1-c_t)^{(1-c_t)(c_r n) + \frac{1}{2}} ((1-c_r)n)^{1/2} c_t^{c_t((1-c_r)n) + \frac{1}{2}} (1-c_t)^{(1-c_t)((1-c_r)n) + \frac{1}{2}}}\right)$$

628 Simplifying this expressoin, we get: $\Theta\left((nc_t(1-c_t)c_r(1-c_r))^{-1/2}\right) = \Theta_1(1/\sqrt{n})$. \square

Lemma E.3. Fix any r, t, n such that $r, t \leq n$. Then,

$$\arg \max_{k \in [n]} \frac{\binom{r}{k} \cdot \binom{n-r}{t-k}}{\binom{n}{t}} = \left\lfloor t \cdot \frac{r}{n} \right\rfloor \text{ or } \left\lceil t \cdot \frac{r}{n} \right\rceil.$$

629 *Proof.* The proof is again simpler if we assume $k = tr/n$ is an integer. If not, the same argument
630 works controlling the errors. In that case, let $k_1 = tr/n + i$ and $k_2 = tr/n + i + 1$ and as before, let
631 $r = c_r n$ and $t = c_t n$. Note that

$$\frac{\binom{r}{k_1} \cdot \binom{n-r}{t-k_1}}{\binom{n}{t}} = \frac{\binom{r}{k_1} \cdot \binom{n-r}{t-k_1}}{\binom{r}{k_2} \cdot \binom{n-r}{t-k_2}} = \frac{\binom{c_r n}{c_t c_r n + i} \cdot \binom{(1-c_r)n}{(1-c_r)c_t n - i}}{\binom{c_r n}{c_t c_r n + i + 1} \cdot \binom{(1-c_r)n}{(1-c_r)c_t n - i - 1}} = \frac{(c_t c_r n + i + 1) \cdot ((1-c_t)(1-c_r)n + i + 1)}{((1-c_t)c_r n - i) \cdot ((1-c_r)c_t n - i)}$$

632 If $i \geq 0$, then the last term is at least $\frac{c_t c_r n \cdot ((1-c_t)(1-c_r)n)}{((1-c_t)c_r n) \cdot ((1-c_r)c_t n)}$ which is greater than one. If $i \leq -1$,
633 then the last term is $\frac{c_t c_r n \cdot ((1-c_t)(1-c_r)n)}{((1-c_t)c_r n) \cdot ((1-c_r)c_t n)}$ which is smaller than one.

634 Thus, $\frac{\binom{r}{k} \cdot \binom{n-r}{t-k}}{\binom{n}{t}}$ is increasing as k increases up to tr/n and then decreases. Thus, the maximum is
635 reached at tr/n . \square

Proof of Lemma 4.1. We first use a simple counting argument to obtain an expression for $\Pr(\mathcal{E}_{k,t,r,n})$ as a ratio of binomial coefficients. We note that there are $\binom{r}{k}$ collections of k red balls, $\binom{n-r}{t-k}$ collections of $t-k$ blue balls, and that the total number of collections of t balls is $\binom{n}{t}$. Since the t balls are drawn uniformly at random without replacement, we get

$$\Pr(\mathcal{E}_{k,t,r,n}) = \frac{\binom{r}{k} \cdot \binom{n-r}{t-k}}{\binom{n}{t}}.$$

636 The $O(1/\sqrt{n})$ bound now follows directly from Lemma E.2 and Lemma E.3. \square

637 Next, we prove Lemma 4.3.

638 **Lemma 4.3.** Fix any deterministic algorithm \mathcal{A} . For any $t = \Theta_1(n)$, $\mathbb{E}[\text{cost}_{\mathcal{A}}(t)] = \Omega(\sqrt{n})$.

Proof. Let σ be a permutation of $[t]$ and \mathcal{O}_{σ} the event that $a_{\sigma(1)} < a_{\sigma(2)} < \dots < a_{\sigma(t)}$. For any deterministic algorithm \mathcal{A} , the choice of the position $\pi(a_t)$ where to place the t -th element depends only on σ . Let $k = \sigma^{-1}(t)$ be the relative rank of the t -th element. Since the distribution of k is uniform in $[t]$ (see the proof of Proposition 3.3), then we have that:

$$\Pr\left[\frac{t}{4} \leq k \leq \frac{3t}{4}\right] = \frac{1}{2}$$

Conditioned on that event $k = \Theta_1(t)$ so we are in the conditions of Lemma 4.2. Therefore, the probability of each rank given the observations is at most $O(1/\sqrt{n})$. Therefore, there is a constant c such that:

$$\Pr\left[|\text{rk}(a_{(t)}) - \pi(a_{(t)})| < c\sqrt{n} \mid \frac{t}{4} \leq k \leq \frac{3t}{4}\right] \leq \frac{1}{2}$$

639 Finally, we observe that:

$$\begin{aligned} \mathbb{E}[\text{cost}_{\mathcal{A}}(t)] &\geq \frac{1}{2} \cdot \mathbb{E}\left[|\text{rk}(a_{(t)}) - \pi(a_{(t)})| \mid \frac{t}{4} \leq k \leq \frac{3t}{4}\right] \\ &\geq \frac{1}{2} \cdot c\sqrt{n} \cdot \Pr\left[|\text{rk}(a_{(t)}) - \pi(a_{(t)})| \geq c\sqrt{n} \mid \frac{t}{4} \leq k \leq \frac{3t}{4}\right] \geq \frac{c\sqrt{n}}{4}. \end{aligned}$$

640 \square

641 We are now ready to prove Theorem 2.

642 *Proof of Theorem 2.* For any deterministic algorithm, sum the bound in Lemma 4.3 for $\Theta(n)$ time
643 steps. For randomized algorithms, the same bound extends via Yao's minimax principle. The reason
644 is that a randomized algorithm can be seen as a distribution on deterministic algorithms parametrized
645 by the random bits it uses. If a randomized algorithm obtains less than $O(n\sqrt{n})$ inversions in
646 expectation, then it should be possible to fix the random bits and obtain a deterministic algorithm
647 with the same performance. \square

648 F Hardness of Online Ranking with Adversarial Ordering

649 **Proposition F.1.** If the ordering σ of the arrival of elements is adversarial, then any algorithm has
650 cost $\Omega(n^2)$ in expectation.

651 *Proof.* At a high level, we construct an ordering such that at each iteration, the arrived element is
652 either the largest or smallest element not yet observed with probability $1/2$ each. Since the algorithm
653 cannot distinguish between the two cases, it suffers a linear cost in expectation at each arrival.

654 Formally, we define σ inductively. At round t , let $i_{t,-}$ and $i_{t,+}$ be the minimum and maximum
655 indices of the elements arrived previously. We define $\sigma(t)$ such that $\sigma(t) = a_{i_{t,-}+1}$ with probability
656 $1/2$ and $\sigma(t) = a_{i_{t,+}-1}$ with probability $1/2$. Thus, the t th element arrived is either the smallest or
657 largest element not yet arrived.

658 The main observation is that the pairwise comparisons at time t are identical whether $a_{(t)} = a_{i_{t,-}+1}$
659 or $a_{(t)} = a_{i_{t,+}-1}$. This is since all the elements previously arrived are either maximal or minimal
660 and there is no elements that are between $a_{i_{t,-}+1}$ and $a_{i_{t,+}-1}$ that have previously arrived. Thus the
661 decision of the algorithm is *independent* of the randomization of the adversary for the t th element.
662 Thus for any learned rank at time t , in expectation over the randomization of the adversary for the
663 element arrived at time t , the learned rank is at expected distance of the true rank at least $n/4$ for
664 $t \leq n/2$. Thus the total cost is $\Omega(n^2)$ in expectation. \square