Algorithms and Data Structures

F. Dehne J.-R. Sack
N. Santoro (Eds.)

Algorithms and Data Structures

2nd Workshop, WADS '91
Ottawa, Canada, August 14-16, 1991
Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Series Editors

Gerhard Goos
GMD Forschungsstelle
Universität Karlsruhe
Vincenz-Priessnitz-Straße 1
W-7500 Karlsruhe, FRG

Juris Hartmanis
Department of Computer Science
Cornell University
Upson Hall
Ithaca, NY 14853, USA

Volume Editors

Frank Dehne
Jörg-Rüdiger Sack
Nicola Santoro
School of Computer Science, Carleton University
Ottawa, Canada K1S 5B6

CR Subject Classification (1991): F.1-2, G.2-3, H.3, I.3.5

ISBN 3-540-54343-0 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-54343-0 Springer-Verlag New York Berlin Heidelberg

Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
2145/3140-543210 - Printed on acid-free paper

PREFACE

The papers in this volume were presented at the Second Workshop on Algorithms and Data Structures (WADS'91). The workshop took place August 14 - 16, 1991, at Carleton University in Ottawa and was organized by the School of Computer Science at Carleton University (Ottawa, Ont). The workshop alternates with the Scandinavian Workshop on Algorithm Theory (SWAT) continuing the tradition of SWAT'88, WADS'89, and SWAT'90.

In response to the program committee's call for papers, 107 papers were submitted. From these submissions, the program committee selected 38 for presentation at the workshop. In addition to these papers, the workshop included five invited presentations.

August 1991

Frank Dehne
Jörg-Rüdiger Sack
Nicola Santoro

PROGRAM COMMITTEE:
A. Apostolico (Purdue U./Aquila U.)
M. Atallah (Purdue U.)
J.-D. Boissonat (INRIA, Sophia Antipolis)
S. Carlson (Lund U.)
K. Clarkson (AT&T Bell Labs.)
F. Dehne (Carleton U.)
J. Gilbert (XEROX, Palo Alto)
M. Goodrich (Johns Hopkins U.)
S. Hambrusch (Purdue U.)
M. Keil (U. of Saskatchewan, Saskatoon)
M. C. Loui (NSF)
F. Meyer auf der Heide (Paderborn)
J.-R. Sack (Carleton U.)
N. Santoro (Carleton U.)
R. Seidel (U. of Berkeley)
R. Tamassia (Brown U.)
N. M. Thalmann (U. of Geneva)
J. Urrutia (Ottawa U.)
J. van Leeuwen (U. of Utrecht)
C.K. Wong (IBM, Yorktown Heights)
D. Wood (U. of Waterloo)

ORGANIZING COMMITTEE:
F. Fiala (Carleton U., Chair)
R. Carter (Carleton U.)
E. Otoo (Carleton U.)
R. Probert (Ottawa U.)

SPONSORED BY
NSERC, ITRC, OCRI, TRIO,
Carleton U. and Queen's U.

TABLE OF CONTENTS

Session 8B
Chaired by E. Otoo

Session 9A
Chaired by M. Goodrich

Session 9B
Chaired by J. Gilbert

Session 10
Invited Presentation

Session 11A
Chaired by R. Tamassia

Session 11B
Chaired by S. Carlson

A Case Study in Comparison Based Complexity: Finding the Nearest Value(s)

Walter Cunto
Centro Científico IBM de Venezuela
A.P. 64778, Caracas, Venezuela

J. Ian Munro
Department of Computer Science, University of Waterloo
Waterloo Ont., N2L 3G1, Canada

Patricio V. Poblete
Departamento de Ciencias de la Computación, Universidad de Chile
Blanco Encalada 2120, Casilla 2777, Santiago, Chile

Abstract. It is shown that $5n/4$ plus-minus lower order terms comparisons on average are necessary and sufficient to solve the problem of finding the values of ranks immediately above and below a specified element x in a set X of size $n > 1$. When x turns out to be the median of X, $1.5n + \sqrt{\pi n/8} + O(\lg n)$ comparisons are proven to be sufficient. $n + \min(k, n-k) + 3\ln n + O(1)$ comparisons are sufficient if k, the rank of x in X, differs from $n/2$ by $\Theta(n)$.

1 Introduction

An interesting although surprisingly little studied problem in selection is that of determining the nearest value in an unordered array to a given value under a pure comparison based model of computation. We address the average case complexity of this problem more formally given as:

Problem 1. *Given a set X of $n > 1$ elements, including a designated $x \in X$, find the elements of ranks one above and one below x; or report the absence of one of these.*

Two variants of the previous problem are also useful. The *left neighbor* problem is that of finding the largest element in X that is less than x if such element exists; otherwise, reporting its absence. The *right neighbor* problem is defined symmetrically. The worst case complexity of this problem has been fully studied [2]. A simple algorithm making at most $2n - 3$ comparisons has shown to be optimal in that case. Worst case optimality is proven by designing a simple adversary which resembles the one given in [1] for the worst case selection problem.

Assuming all input permutations are equally likely, a somewhat faster method, on average, can be demonstrated. This algorithm performs $1.5n + \sqrt{\pi n/8} + O(\lg n)$ comparisons if x turns out to be (virtually) the median of X; otherwise, it performs $n + \min(k, n-k) + 3\ln n + O(1)$ comparisons where k denotes the rank of x in X. If any of the possible ranks of x in X is also equally likely, an average of $5n/4 + H_n/2 + 11H_{\lfloor n/2 \rfloor}/4 + O(1)$ comparisons are performed by the algorithm.

These estimates of runtime are derived with the help of a Markovian graph model wherein nodes represent computational states and edges represent transitions among states performed during the computation of any problem instance. Computations are traced by traversing paths in the graph and average performances are obtained by counting average costs (comparisons) associated either to the nodes or to the edges of the graph. Edge oriented counts along the traversals were used to derive the performance of our algorithm conditioned to the (previously unknown) rank of x in X. Node oriented counts were used to derive a closed formula for the average performance of our algorithm. Finally, $5n/4 - \Omega(1)$ comparisons are shown to be required by any algorithm that solves the problem with a technique slightly different than that discussed in [3] which counts different types of comparisons along the computation of the solution.

2 The algorithm

The algorithm keeps track of closest neighbors found thus far on either side of x, together with a count on the number elements seen on each side. More formally:

i) Compare the first element with x, making it a neighbor candidate on the appropriate side.

ii) Process each remaining element by comparing it with the current neighbor on the more populous side of x. In case of equal population, the neighbor is randomly chosen.

iii) If necessary, compare the new element with the other neighbor.

iv) If the new element falls between the current neighbors, compare it with x and replace the appropriate neighbor candidate with the new value.

This algorithm, which is also suitable for on-line applications, performs at most $3n-6$ comparisons; but, as we shall see, its average case behavior is more interesting.

3 A Markovian graph model

At each step, the algorithm determines whether the next element is larger or smaller than x. This process is modeled as a Markovian graph $G = (V, E)$. V includes all states $\langle p, q\rangle$ with $p \geq 0$, $q \geq 0$ and $p + q \leq n - 1$ such that p and q are the numbers of elements smaller and larger than x respectively after $p + q$ steps. E contains all possible state transitions. The collection of subsets $V_t = \{\langle p, q\rangle \mid p + q = t - 1\}$, $1 \leq t \leq n$, partitions the set V. Clearly, $|V_t| = t$ and $|V| = \binom{n+1}{2}$ as transitions occur only from nodes in V_{t-1} to nodes in V_t, $2 \leq t \leq n$. The computation starts at $s = \langle 0, 0\rangle$ and finishes at any of the n states $\langle p, q\rangle$ such that $p + q = n - 1$ with $p, q \geq 0$.

A directed edge is denoted by (v, w, j) where $v \in V_{t-1}$, $w \in V_t$, $2 \leq t \leq n$, and $j \in E(v, w)$ is the label of one of the transitions from v to w. Each edge $(v, w, j) \in E$ specifies the number of comparisons $c(v, w, j)$ to be executed and its transition probability $pr(v, w, j)$. Note that G is nonsimple.

Since G is a Markovian graph, the sum of probabilities associated with transitions starting from the same node must equal 1, that is

$$\forall v \in V_{t-1},\ 2 \leq t \leq n,\ \sum_{w \in V_t} \sum_{j \in E(v,w)} pr(v, w, j) = 1\ . \tag{1}$$

Also, transitions in G are symmetric with respect to central nodes (those states $\langle p, q\rangle$ such that $|p - q| \leq 1$). Figure 1 summarizes the number of comparisons and the transition

First type of transitions, $p = q = 0$,

$$c(\langle 0,0\rangle, \langle 0,1\rangle, 1) = c(\langle 0,0\rangle, \langle 1,0\rangle, 1) = 1$$

$$pr(\langle 0,0\rangle, \langle 0,1\rangle, 1) = pr(\langle 0,0\rangle, \langle 1,0\rangle, 1) = \tfrac{1}{2}$$

Second type of transitions, $\min(p,q) = 0$ and $\max(p,q) > 0$

$$\begin{array}{rclcl}
c(\langle p,q\rangle, \langle p,q+1\rangle, 1) & = & c(\langle q,p\rangle, \langle q+1,p\rangle, 1) & = & 1 \\
c(\langle p,q\rangle, \langle p,q+1\rangle, 2) & = & c(\langle q,p\rangle, \langle q+1,p\rangle, 2) & = & 2 \\
c(\langle p,q\rangle, \langle p+1,q\rangle, 1) & = & c(\langle q,p\rangle, \langle q,p+1\rangle, 1) & = & 2
\end{array}$$

$$\begin{array}{rclcl}
pr(\langle p,q\rangle, \langle p,q+1\rangle, 1) & = & pr(\langle q,p\rangle, \langle q+1,p\rangle, 1) & = & \frac{p+q}{p+q+2} \\
pr(\langle p,q\rangle, \langle p,q+1\rangle, 2) & = & pr(\langle q,p\rangle, \langle q+1,p\rangle, 2) & = & \frac{1}{p+q+2} \\
pr(\langle p,q\rangle, \langle p+1,q\rangle, 1) & = & pr(\langle q,p\rangle, \langle q,p+1\rangle, 1) & = & \frac{1}{p+q+2}
\end{array}$$

Third type of transitions, $p, q > 0$ and $p \neq q$,

$$\begin{array}{rclcl}
c(\langle p,q\rangle, \langle p,q+1\rangle, 1) & = & c(\langle q,p\rangle, \langle q+1,p\rangle, 1) & = & 1 \\
c(\langle p,q\rangle, \langle p,q+1\rangle, 2) & = & c(\langle q,p\rangle, \langle q+1,p\rangle, 2) & = & 3 \\
c(\langle p,q\rangle, \langle p+1,q\rangle, 1) & = & c(\langle q,p\rangle, \langle q,p+1\rangle, 1) & = & 2 \\
c(\langle p,q\rangle, \langle p+1,q\rangle, 2) & = & c(\langle q,p\rangle, \langle q,p+1\rangle, 2) & = & 3
\end{array}$$

$$\begin{array}{rclcl}
pr(\langle p,q\rangle, \langle p,q+1\rangle, 1) & = & pr(\langle q,p\rangle, \langle q+1,p\rangle, 1) & = & \frac{\max(p,q)}{p+q+2} \\
pr(\langle p,q\rangle, \langle p,q+1\rangle, 2) & = & pr(\langle q,p\rangle, \langle q+1,p\rangle, 2) & = & \frac{1}{p+q+2} \\
pr(\langle p,q\rangle, \langle p+1,q\rangle, 1) & = & pr(\langle q,p\rangle, \langle q,p+1\rangle, 1) & = & \frac{\min(p,q)}{p+q+2} \\
pr(\langle p,q\rangle, \langle p+1,q\rangle, 2) & = & pr(\langle q,p\rangle, \langle q,p+1\rangle, 2) & = & \frac{1}{p+q+2}
\end{array}$$

Fourth type of transitions, $p = q > 0$,

$$\begin{array}{rclcl}
c(\langle p,p\rangle, \langle p,p+1\rangle, 1) & = & c(\langle p,p\rangle, \langle p+1,p\rangle, 1) & = & 1 \\
c(\langle p,p\rangle, \langle p,p+1\rangle, 2) & = & c(\langle p,p\rangle, \langle p+1,p\rangle, 2) & = & 2 \\
c(\langle p,p\rangle, \langle p,p+1\rangle, 3) & = & c(\langle p,p\rangle, \langle p+1,p\rangle, 3) & = & 3
\end{array}$$

$$\begin{array}{rclcl}
pr(\langle p,p\rangle, \langle p,p+1\rangle, 1) & = & pr(\langle p,p\rangle, \langle p+1,p\rangle, 1) & = & \frac{p/2}{2p+2} \\
pr(\langle p,p\rangle, \langle p,p+1\rangle, 2) & = & pr(\langle p,p\rangle, \langle p+1,p\rangle, 2) & = & \frac{p/2}{2p+2} \\
pr(\langle p,p\rangle, \langle p,p+1\rangle, 3) & = & pr(\langle p,p\rangle, \langle p+1,p\rangle, 3) & = & \frac{1}{2p+2}
\end{array}$$

Fig. 1. Summary of comparisons and probabilities per type of transition

probabilities associated with edges in the graph. The probability value of each transition follows from the assumption that any permutation of the input data is equally likely.

The computation of any given input instance is traced by a path starting from s and ending at one of the nodes in V_n. Transitions in the path follow an increasing sequence according to the partition of V and different instances may follow the same path. For a given instance, the number of comparisons performed is the sum of comparisons of each edge along the path followed. The probability of traversing any path is the product of probabilities of each edge in it.

Let $v \to w$ denote any possible transition between two designated nodes and $s \xrightarrow{*} w$, any path from the initial node to a node w. The average number of comparisons performed by the algorithm with an input of size n is

$$C_n = \sum_{\substack{w \in V_n \\ s \xrightarrow{*} w}} c(s \xrightarrow{*} w) pr(s \xrightarrow{*} w) . \tag{2}$$

The probability of reaching a node $w \in V$ is

$$pr(w) = \sum_{s \xrightarrow{*} w} pr(s \xrightarrow{*} w) \tag{3}$$

and the probability that the algorithm performs a transition in $E(v,w)$ is given by

$$pr(v,w) = pr(v) \left(\sum_{j \in E(v,w)} pr(v,w,j) \right) . \tag{4}$$

Equations (3) and (4) are related. A simple induction on the path sequence shows that

$$pr(w) = \sum_{v \in V} pr(v,w) . \tag{5}$$

The average cost associated with each vertex $v \in V$ and the average cost associated with each set of transitions $E(v,w)$ are defined respectively as

$$\bar{c}(v) = \sum_{w \in V} \sum_{j \in E(v,w)} c(v,w,j) pr(v,w,j) \tag{6}$$

and

$$\bar{c}(v,w) = \frac{\sum_{j \in E(v,w)} c(v,w,j) pr(v,w,j)}{\sum_{j \in E(v,w)} pr(v,w,j)} . \tag{7}$$

The following lemma presents two methods for computing the average number of comparisons C_n. The first one is *node-oriented* while the second is *edge-oriented.* In addition, both methods can be adapted to any dynamic process described by an acyclic Markovian graph with transition costs.

Lemma 1. *The average number of comparisons C_n performed by the algorithm can is*

$$C_n = \sum_{v \in V} \bar{c}(v) pr(v) = \sum_{v,w \in V} \bar{c}(v,w) pr(v,w) .$$

Proof. The proof is by induction on n which trivially holds for $n = 1$. When $n > 1$ and transitions from V_{n-1} to V_n are fixed, equation (2) can be rewritten as

$$C_n = \sum_{v \in V_{n-1}} \sum_{w \in V_n} \sum_{s \xrightarrow{*} v \to w} c(s \xrightarrow{*} v \to w) pr(s \xrightarrow{*} v \to w) .$$

Grouping all possible transitions from v to w gives

$$C_n = \sum_{v \in V_{n-1}} \sum_{s \xrightarrow{*} v} \sum_{w \in V_n} \sum_{j \in E(v,w)} (c(s \xrightarrow{*} v) + c(v,w,j)) pr(s \xrightarrow{*} v) pr(v,w,j) . \tag{8}$$

$$\bar{c}(\langle p,q\rangle) = \begin{cases} 1 & \text{if } p=q=0, \\ 1+\frac{2}{p+q+2} & \text{if } \min(p,q)=0 \text{ and } \max(p,q)>0, \\ 1+\frac{\min(p,q)+4}{p+q+2} & \text{if } p,q>0 \text{ and } p\neq q, \\ \frac{3}{2}+\frac{3}{2p+2} & \text{if } p=q>0. \end{cases}$$

Fig. 2. Summary of average cost by type of state

From (1), equation (8) becomes

$$\begin{aligned} C_n = & \sum_{\substack{v\in V_{n-1} \\ s \xrightarrow{*} v}} c(s\xrightarrow{*} v)pr(s\xrightarrow{*} v) \\ & + \sum_{v\in V_{n-1}} \left(\sum_{s\xrightarrow{*} v} pr(s\xrightarrow{*} v)\right)\left(\sum_{w\in V_n}\sum_{j\in E(v,w)} c(v,w,j)pr(v,w,j)\right) . \end{aligned} \tag{9}$$

When (2), (3) and (6) are taken into consideration,

$$C_n = C_{n-1} + \sum_{v\in V_{n-1}} pr(v)\bar{c}(v) . \tag{10}$$

Otherwise, if (2), (4) and (7) are substituted into (9),

$$C_n = C_{n-1} + \sum_{\substack{v\in V_{n-1} \\ w\in V_n}} pr(v,w)\bar{c}(v,w) . \tag{11}$$

The lemma follows by carrying forward the inductive hypothesis. ∎

Since equations (10) and (11) are recurrent, the average number of comparisons C_n can be easily computed.

4 Average case upper bounds

When the execution of an instance is traced with a Markovian graph described above, any of the nodes in the same partition subset is equally likely to be reached by the algorithm. This property is stated in the next lemma.

Lemma 2. *$\forall w\in V_t$, $1\leq t\leq n$, $pr(w)=1/t$.*

Sketch of proof. From (4) and (5), the probability of reaching any node in V_t can be inductively defined as

$$pr(w) = \sum_{v\in V_{t-1}} pr(v) \sum_{j\in E(v,w)} pr(v,w,j) .$$

A proof by cases with the cases presented in Figure 1 completes the proof of the lemma. ∎

Corollary 3.

$$C_n = \sum_{1\leq t\leq n-1} \frac{1}{t} \sum_{w\in V_t} \bar{c}(w) . \tag{12}$$

Figure 2 displays the average cost for type of nodes in the graph.

Theorem 4. *If $|X| = n \geq 1$, the average number of comparisons performed by the algorithm to find both neighbors of $x \in X$ is*

$$C_n = \begin{cases} 0 & \text{if } n = 0, \\ 1 & \text{if } n = 1, \\ \frac{8}{3} & \text{if } n = 3, \text{ and} \\ \frac{5}{4}n + \frac{1}{2}H_n + \frac{11}{4}\left(H_{\lfloor n/2 \rfloor} - \frac{n \bmod 2}{2}\right) - \frac{27}{4} + \frac{4}{n} & \text{if } n \geq 4. \end{cases}$$

where $H_n = \sum_{i=1}^{n} 1/i = \ln n + O(1)$.

Proof. Regrouping equation (12) by type of nodes,

i) $n \geq 5$.

$$C_n = C_{n-1} + \frac{2}{n-1}\left(\frac{n+2}{n} + \sum_{1 \leq p \leq \lfloor n/2 \rfloor - 2} \frac{n+p+4}{n} + \frac{3n+6}{4n}((n+1) \bmod 2)\right)$$

ii)

$$C_1 = 0,\ C_2 = 1,\ C_3 = \frac{8}{3} \text{ and } C_4 = \frac{25}{6}.$$

Algebraic manipulation of the previous equation leads us to

i) odd $n \geq 7$

$$C_n = C_{n-2} + \frac{5}{2} + \frac{29n}{4} + \frac{6}{n-1} - \frac{27}{4(n-2)},$$

ii) even $n \geq 6$

$$C_n = C_{n-2} + \frac{5}{2} + \frac{10}{n} + \frac{1}{2(n-1)} - \frac{4}{n-2}.$$

The desired result stated above is obtained by recurring on n. ∎

An interesting question is how many comparisons are performed on the average if the rank of x in X turns out to be k. In this case, we will show that $n + \min(k, n-k) + o(n)$ comparisons suffice. Moreover, when x happens to be the median of X, the lower order term becomes $O(\sqrt{n})$, as the algorithm is essentially betting the new element will fall on the less likely side of x.

Let us consider the algorithm starting at state s and stopping when some predefined state $\langle p, q \rangle$ is reached. As explained before, each possible execution of the program determines a path from $\langle 0, 0 \rangle$ to $\langle p, q \rangle$ and since we are sampling without replacement, each one of the $(p, q) = \binom{p+q}{p}$ paths is equally likely.

Conditioned to the fact that $\langle p, q \rangle$ is the final state, the probability that any of the possible transitions between two states is traversed by the algorithm is

$$pr(\langle i, j \rangle, \langle i+1, j \rangle) = \frac{(i,j)(p-i-1, q-j)}{(p,q)} \text{ and}$$

$$pr(\langle i, j \rangle, \langle i, j+1 \rangle) = \frac{(i,j)(p-i, q-j-1)}{(p,q)}.$$

Such probabilities are zero if the corresponding transitions are not included in any of the paths between the initial and the fixed final state.

$$\bar{c}(\langle i,j\rangle \rightarrow \langle i,j+1\rangle) = \begin{cases} 1 & \text{if } i=j=0, \\ 1+\frac{1}{j+1} & \text{if } i=0 \text{ and } j>0, \\ 2 & \text{if } i>0 \text{ and } j=0, \\ 2+\frac{1}{j+1} & \text{if } i>0 \text{ and } 0<j<i, \\ \frac{3}{2}+\frac{3/2}{j+1} & \text{if } i>0 \text{ and } j=i, \\ 1+\frac{2}{j+1} & \text{if } i>0 \text{ and } j>i. \end{cases}$$

Fig. 3. Summary of average cost per type of grouped transitions

The average performance $C(p,q)$ will be computed with equation (11) adapted to this specific context. Observe that the subgraph associated with the execution of the algorithm will be confined within states $\langle i,j\rangle$ such that $0 \leq i \leq p$ and $0 \leq j \leq q$. The average cost per type of transition is given in Figure 3. Such average costs are symmetric, that is

$$\bar{c}(\langle i,j\rangle \rightarrow \langle i+1,j\rangle) = \bar{c}(\langle j,i\rangle \rightarrow \langle j,i+1\rangle) .$$

Theorem 5. *The average number of comparisons performed by the algorithm conditioned to the fact that it stops at state $\langle p,q\rangle$ is*

$$\begin{aligned} C(p,q) &= \max(p,q) + 2\min(p,q) + 2H_{\max(p,q)} + H_{\min(p,q)} - 2H_{p+q} + H_p + H_q - 2 \\ &\quad + \delta_{p,0} + \delta_{q,0} - \frac{1}{2}\left(1-\frac{1}{p+1}\right)\delta_{p,q} \\ &\quad + \frac{1}{2}\sum_{0\leq j\leq \min(p,q)} \left(1-\frac{1}{j+1}\right)\frac{(j,j)(p-j,q-j)}{(p,q)} . \end{aligned} \tag{13}$$

Proof.

$$\begin{aligned} C(p,q) &= \sum_{j\geq 0}\left(1+\frac{2}{j+1}\right)\sum_{0\leq i\leq j} pr(\langle i,j\rangle,\langle i,j+1\rangle) \\ &\quad + \sum_{i\geq 0}\left(1+\frac{2}{i+1}\right)\sum_{0\leq j\leq i} pr(\langle i,j\rangle,\langle i+1,j\rangle) \\ &\quad + \sum_{j\geq 0}\left(2+\frac{1}{j+1}\right)\sum_{i>j} pr(\langle i,j\rangle,\langle i,j+1\rangle) + \sum_{i\geq 0}\left(2+\frac{1}{i+1}\right)\sum_{j>i} pr(\langle i,j\rangle,\langle i+1,j\rangle) \\ &\quad + \frac{1}{2}\sum_{j\geq 0}\left(1-\frac{1}{j+1}\right)\left(pr(\langle j,j\rangle,\langle j,j+1\rangle) + pr(\langle j,j\rangle,\langle j+1,j\rangle)\right) \\ &\quad - \sum_{j\geq 0}\frac{1}{j+1} pr(\langle 0,j\rangle,\langle 0,j+1\rangle) - \sum_{i\geq 0}\frac{1}{i+1} pr(\langle i,0\rangle,\langle i+1,0\rangle) \\ &\quad - \sum_{j\geq 0} pr(\langle 0,j\rangle,\langle 1,j\rangle) - \sum_{i\geq 0} pr(\langle i,0\rangle,\langle i,1\rangle) . \end{aligned}$$

Simplification of the inner summations gives

$$C(p,q) = \sum_{0\le j\le \max(p,q)} \left(1+\frac{2}{j+1}\right) + \sum_{0\le j\le \min(p,q)} \left(2+\frac{1}{j+1}\right)$$
$$+\frac{1}{2} \sum_{\substack{0\le j\le \min(p,q)\\ j<\max(p,q)}} \left(1-\frac{1}{j+1}\right) \frac{(j,j)(p-j,q-j)}{(p,q)} - \sum_{0\le j<q} \frac{1}{j+1}\frac{(p,q-j-1)}{(p,q)}$$
$$-\sum_{0\le i<p} \frac{1}{i+1}\frac{(p-i-1,q)}{(p,q)} - \sum_{0\le j\le q} \frac{(p-1,q-j)}{(p,q)}(1-\delta_{p,0})$$
$$-\sum_{0\le i\le p} \frac{(p-i,q-1)}{(p,q)}(1-\delta_{q,0}) \; .$$

Expression (13) is obtained by using identities A1 and A2 from the Appendix in the previous expression. ∎

Theorems 4 and 5 are related since it is not difficult to realize that

$$C_n = 2/n \sum_{0\le j\le \lceil n/2\rceil -1} C(j,n-1-j) \; .$$

From Theorem 5, two particular cases are considered:
i) $p=q=(n+1)/2$ with odd n,
ii) $p=\alpha n$, $q=(1-\alpha)n$ for a fixed $\alpha\in(0,\frac{1}{2})$.
The case $\alpha\in(\frac{1}{2},1)$ is symmetric to *ii*.

Theorem 6. *The asymptotic average number of comparisons performed by the algorithm when x is the median of X (and n is odd) is*

$$C\left(\frac{n+1}{2},\frac{n+1}{2}\right) = \frac{3}{2}n+\frac{1}{2}\sqrt{\frac{\pi n}{2}}+3\ln n+O(1) \; .$$

Proof. If p is substituted for q into equation (13),

$$C(p,p) = 3p+5H_p-2H_{2p}-\frac{5}{2}+\frac{1}{2(p+1)}+2\delta_{p,0}+\frac{1}{2}\sum_{0\le j\le p}\left(1-\frac{1}{j+1}\right)\frac{(j,j)(p-j,p-j)}{(p,p)} \; .$$

Further simplification of the previous expression is obtained with identities A3 and A4 from the Appendix, that is

$$C(p,p) = 3p+5H_p-2H_{2p}-\frac{5}{2}+2\delta_{p,0}-\frac{p}{1+p}+\frac{1}{2}\frac{4^p}{(p,p)} \; .$$

The previous equation is then asymptotically expanded with identities A6 and A8 from the Appendix getting

$$C(p,p) = 3p+\frac{1}{2}\sqrt{\pi p}+3\ln n+3\gamma-\frac{7}{2}-2\ln 2+\frac{1}{16}\sqrt{\frac{\pi}{p}}+\frac{3}{p}+\frac{1}{256}\sqrt{\frac{\pi}{p^3}}-\frac{11}{8}\frac{1}{p^2}+\cdots$$

A simple replacement of p by $(n+1)/2$ proves the lemma. ∎

Theorem 7. *The average number of comparisons performed by the algorithm when the rank of x in X is αn, for any fixed $\alpha < 1/2$, is*

$$C(\alpha n, (1-\alpha n)) = (1+\alpha)n + 3\ln n + O(1) .$$

Proof. If $p = \alpha n$ and $q = (1-\alpha)n$ in equation (13) and identity A5 from the Appendix is applied, then

$$C(\alpha n, (1-\alpha n)) = (1+\alpha)n + 2H_{\alpha n} + 3H_{(1-\alpha)n} - 2H_n - 2$$
$$+ \frac{1}{2}\frac{\alpha n}{(1-\alpha)n+1}\sum_{j\geq 0}\frac{(\alpha n)^{\underline{j}}}{n^{\underline{j}}}2^k - \frac{1}{2}\frac{1}{(1-\alpha)n+1}\sum_{j\geq 0}\frac{(\alpha n)^{\underline{j}}}{n^{\underline{j}}}k2^k .$$

Identity A8 from the Appendix with $g(x) = 1/(1-2x)$ and $g(x) = 2x/(1-2x)$ solves the summations in the previous expressions. ∎

5 Average case lower bounds

Let us consider the following (easier) problem:

Problem 2. *X is a set of $n > 1$ numbers with two designated neighbors w and x, such that $w < x$, verify that w and x are indeed of consecutive ranks (ranks of w and x are unknown in advance).*

It is assumed that elements w and x are stored in registers, while the other $n-2$ elements are in the array X. To simplify the discussion, we will assume the values in question constitute the distinct integers 1 through n with w and x of consecutive but unknown ranks; thus, there is no need to distinguish between the kth smallest element of X and the number k. Clearly, any lower bound for Problem 2 is also one for the more general Problem 1.

In developing a lower bound for the number of comparisons to be performed by any algorithm which solves Problem 2, three types of comparisons will be considered: *partition comparisons, straddle comparisons* and *closer comparisons.* Any solution of Problem 2 must identify the elements smaller and larger than x. Thus, for any element smaller than w, its *partition comparison* is the first comparison between it and either w or another element lying between these two. Symmetrically, if the element is greater than x, its partition comparison is the first comparison between it and either x or an intermediate element between both elements. It is not difficult to realize that $n-2$ of such comparisons must be performed in order to get a consistent solution. It is expected, however, that some comparisons which are not partition comparisons will be performed, and in this case, we will focus only on *straddle comparisons*: those involving an element not greater than w with another not smaller than x.

Partition and straddle comparisons will be related through the concept of *closer comparison.* Let $\theta_\pi(w)$ and $\theta_\pi(x)$ denote the rank of w and x in X for a given the input permutation π respectively such that $\theta_\pi(x) = \theta_\pi(w) + 1$. The closer comparison of any element $k \in X$ for a given π is the first comparison between it and an element $l \in X$ subject to

i) $k = \theta_\pi(w) - i$, $i \in [1..\min\{\theta_\pi(w) - 1, n - \theta_\pi(w)\}]$, and $l \in [k+1..\theta_\pi(w) + i]$, or
ii) $k = \theta_\pi(x) + i$, $i \in [1..\min\{\theta_\pi(x) - 1, n - \theta_\pi(x)\}]$, and $l \in [\theta_\pi(x) - i..k-1]$.