9.1-1

Show that the second smallest of n elements can be found with $n + \lceil \lg n \rceil - 2$ comparisons in the worst case. (*Hint:* Also find the smallest element.)

1. 透過兩兩比對找出最小的值 $(n-1)$
2. 記錄比對結果，第二小的值會與最小值經過比對。
3. 從最小值的比對記錄中找出最小 $(\lceil \lg n \rceil - 1)$

example : $\{1, 4, 6, 9, 5, 8, 2, 3\}$

min/max	1	6	5	2
	④	9	8	3

→

min/max	1	2
	⑥	5

→

min/max	1
	②

min : 1 , 2nd smallest = min (4, 6, 2) = 2

9.1-2

Given $n > 2$ distinct numbers, you want to find a number that is neither the minimum nor the maximum. What is the smallest number of comparisons that you need to perform?

1. If n is odd :

$$1 + \frac{3(n-3)}{2} + 2 = \frac{3n}{2} - \frac{3}{2} = \left(\left\lceil \frac{3n}{2} \right\rceil - \frac{1}{2}\right) - \frac{3}{2} = \left\lceil \frac{3n}{2} \right\rceil - 2$$

2. If n is even :

$$1 + \frac{3(n-2)}{2} = \frac{3n}{2} - 2 = \left\lceil \frac{3n}{2} \right\rceil - 2$$ ※

9.2-3

Suppose that RANDOMIZED-SELECT is used to select the minimum element of the array $A = \langle 2, 3, 0, 5, 7, 9, 1, 8, 6, 4 \rangle$. Describe a sequence of partitions that results in a worst-case performance of RANDOMIZED-SELECT.

worst-case : if we keep choosing the maximum

< 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 >

9.3-3

Show how to use SELECT as a subroutine to make quicksort run in $O(n \lg n)$ time in the worst case, assuming that all elements are distinct.

We can use SELECT to choose the median for our pivot. Then the time taken to find the median is on the same order of the rest of the partitioning.

9.3-5

Show how to determine the median of a 5-element set using only 6 comparisons.

1. compare the first and the second two elements (2)
   in the first two, let the smaller be A, the other be B.
   in the second two, let the smaller be C, the other be D.
2. compare B with C (1)
   if $B < C$: $A < B < C < D$, the median could be B, C or E. compare them to get the median. (maximum 2 comparisons)
   if $B > C$: compare A with C and B with D (2)
   if $A < C$: $A < C < B < D$
   $A < C < D < B$ (maximum 2)
   if $A > C$: $C < A < B < D$
   $C < D < A < B$ (maximum 2)

9.3-7

Professor Olay is consulting for an oil company, which is planning a large pipeline running east to west through an oil field of n wells. The company wants to connect a spur pipeline from each well directly to the main pipeline along a shortest route (either north or south), as shown in Figure 9.4. Given the x- and y-coordinates of the wells, how should the professor pick an optimal location of the main pipeline to minimize the total length of the spurs? Show how to determine an optimal location in linear time.

if n is even, then the optimal will be somewhere on or between the lower median and the upper median.

if n is odd, it will be the median.

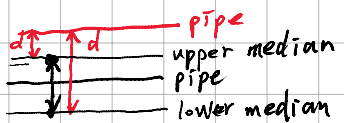
---

if n is even:

if the pipe is between the lower and the upper median = we move the pipeline by distance d then the length of the spurs will be $s' = s + \frac{n}{2}d - \frac{n}{2}d = s$ (both half)

if the pipe is moved beyond one median:

it will increase the total length by $\geq d$



if n is odd:

if we move the pipe d distance away from the median $\frac{(n+1)}{2}$ distance increased, $\frac{(n-1)}{2}$ decreased, $s' = s + d\frac{(n+1)}{2} - d\left(\frac{n-1}{2}\right) = s + d > s$.

a. Argue that the median of $x_1, x_2, \ldots, x_n$ is the weighted median of the x_i with weights $w_i = 1/n$ for $i = 1, 2, \ldots, n$.

b. Show how to compute the weighted median of n elements in $O(n \lg n)$ worst-case time using sorting.

c. Show how to compute the weighted median in $\Theta(n)$ worst-case time using a linear-time median algorithm such as SELECT from Section 9.3.

a.

$$\sum_{x_i < x} w_i = \sum_{x_i < x} \frac{1}{n} = \frac{1}{n} \sum_{x_i < x} 1 \le \frac{1}{n} \cdot \frac{n}{2} = \frac{1}{2}$$

$$\sum_{x_i > x} w_i = \sum_{x_i > x} \frac{1}{n} = \frac{1}{n} \sum_{x_i > x} \frac{1}{n} \le \frac{1}{n} \cdot \frac{n}{2} = \frac{1}{2}$$

b. sort the x_i's and scan throught when the weights' sum exceed $\frac{1}{2}$ we find the median.
sorting : $O(n \lg n)$ scan : $O(n)$
total : $O(n \lg n)$

C. find the median x_k in $X = \{x_1, x_2, \ldots x_n\}$
partition X around x_k in X_L, X_R
if $w_L < \frac{1}{2}$ and $w_R < \frac{1}{2}$ then x_k is median.
else recursively call the half with its sum of weights $\ge \frac{1}{2}$.
$T(n) = T\left(\frac{n}{2} + 1\right) + \Theta(n)$, we recurse once

11.1-1

A dynamic set S is represented by a direct-address table T of length m. Describe a procedure that finds the maximum element of S. What is the worst-case performance of your procedure?

if we have to scan through : $\Theta(m)$

11.1-2

A ***bit vector*** is simply an array of bits (each either 0 or 1). A bit vector of length m takes much less space than an array of m pointers. Describe how to use a bit vector to represent a dynamic set of distinct elements drawn from the set $\{0, 1, \ldots, m-1\}$ and with no satellite data. Dictionary operations should run in $O(1)$ time.

vector[i] = 1 if set has value i

11.2-3

Professor Marley hypothesizes that he can obtain substantial performance gains by modifying the chaining scheme to keep each list in sorted order. How does the professor's modification affect the running time for successful searches, unsuccessful searches, insertions, and deletions?

successful searches, deletions : unaffected.
unsuccessful searches : faster
insertions : slower ($O(n)$)

11.2-5

You need to store a set of n keys in a hash table of size m. Show that if the keys are drawn from a universe U with $|U| > (n-1)m$, then U has a subset of size n consisting of keys that all hash to the same slot, so that the worst-case searching time for hashing with chaining is $\Theta(n)$.

if $|U| = (n-1)m$: each slot has $(n-1)$ keys
if $|U| > (n-1)m$: $\exists$ slot has n keys.

11.3-2

You hash a string of r characters into m slots by treating it as a radix-128 number and then using the division method. You can represent the number m as a 32-bit computer word, but the string of r characters, treated as a radix-128 number, takes many words. How can you apply the division method to compute the hash value of the character string without using more than a constant number of words of storage outside the string itself?

$$h(s) = \sum_{i=1}^{len(s)} (s_i \bmod m)$$

11.3-3

Consider a version of the division method in which $h(k) = k \bmod m$, where $m = 2^p - 1$ and k is a character string interpreted in radix 2^p. Show that if string x can be converted to string y by permuting its characters, then x and y hash to the same value. Give an example of an application in which this property would be undesirable in a hash function.

let x_i be the ith character of x

y_i ' y

$$h(x) - h(y) = \sum_{i=0}^{n-1} x_i 2^{ip} \bmod (2^p-1) - \sum_{i=0}^{n-1} y_i 2^{ip} \bmod (2^p-1)$$

let $x_a = y_b$, $x_b = y_a$, $a \neq b$

$$h(x) - h(y) = \left[(x_a 2^{ap} + x_b 2^{bp}) - (y_a 2^{ap} + y_b 2^{bp}) \right] \bmod (2^p-1)$$

$$= \left[(x_a 2^{ap} + x_b 2^{bp}) - (x_b 2^{ap} + x_a 2^{bp}) \right] \bmod (2^p-1)$$

$$= \left[(x_a - x_b)(2^{ap} - 2^{bp}) \right] \bmod (2^p-1)$$

$$= \left[(x_a - x_b) 2^{bp} (2^{(a-b)p} - 1) \right] \bmod (2^p-1)$$

$$= \sum_{i=0}^{a-b-1} 2^{pi} = \frac{2^{(a-b)p} - 1}{2^p - 1} \quad (r = 2^p)$$

$$\Longrightarrow h(x) - h(y) \bmod (2^p-1) = 0$$

11.3-4

Consider a hash table of size $m = 1000$ and a corresponding hash function $h(k) = \lfloor m\,(kA \bmod 1)\rfloor$ for $A = (\sqrt{5} - 1)/2$. Compute the locations to which the keys 61, 62, 63, 64, and 65 are mapped.

$h(61) = \left\lfloor 1000\left(61 * \frac{\sqrt{5}-1}{2} \bmod 1\right)\right\rfloor = 700$

$h(62) = 339$, $h(63) = 936$, $h(64) = 573$, $h(65) = 192$

11.4-1

Consider inserting the keys $10, 22, 31, 4, 15, 28, 17, 88, 59$ into a hash table of length $m = 11$ using open addressing. Illustrate the result of inserting these keys using linear probing with $h(k, i) = (k + i) \bmod m$ and using double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (k \bmod (m - 1))$.

linear probing

0	1	2	3	4	5	6	7	8	9	10
22	88			4	15	28	17	59	31	10

double hashing

0	1	2	3	4	5	6	7	8	9	10
22		59	17	4	15	28	88		31	10

$h_2(15) = 1 + 15 \bmod 10 = 6$, $(4+6) \bmod 11 = 10$, $(4+12) \bmod 11 = 5$

$h_2(17) = 1 + 17 \bmod 10 = 8$ $(6+8) \bmod 11 = 3$

$h_2(88) = 1 + 88 \bmod 10 = 9$, $(0+9) \bmod 11 = 9$, $(0+18) \bmod 11 = 7$

$h_2(59) = 1 + 59 \bmod 10 = 10$, $(4+10) \bmod 11 = 3$, $(4+20) \bmod 11 = 2$

11.4-4

Show that the expected number of probes required for a successful search when $\alpha = 1$ (that is, when $n = m$), is H_m, the mth harmonic number.

$$E[X] = \frac{1}{m}\sum_{n=1}^{m} E[X_n]$$, let Y_n the expected number of probes, $\alpha = \frac{n}{m}$

$$= \frac{1}{m}\sum_{n=0}^{m-1} E[Y_n] \le \frac{1}{m}\sum_{n=0}^{m-1} \frac{1}{1-\frac{n}{m}}$$

$$= \sum_{n=0}^{m-1} \frac{1}{m-n} = H_m$$ ∎

11-1 Longest-probe bound for hashing

Suppose you are using an open-addressed hash table of size m to store $n \le m/2$ items.

a. Assuming independent uniform permutation hashing, show that for $i = 1, 2, \ldots, n$, the probability is at most 2^{-p} that the ith insertion requires strictly more than p probes.

b. Show that for $i = 1, 2, \ldots, n$, the probability is $O(1/n^2)$ that the ith insertion requires more than $2 \lg n$ probes.

a. $P\{X \ge i\} \le \alpha^{i-1}$, $\alpha = \frac{n}{m}$

$\Rightarrow P\{X > p\} \le \alpha^p = \left(\frac{n}{m}\right)^p \le 2^{-p}$ ∎ $\left(n \le \frac{m}{2}\right)$

b. $2\lg n \in \mathbb{N} \rightarrow P\{X > 2\lg n\} \le 2^{-2\lg n} = n^{-2} = O\left(\frac{1}{n^2}\right)$

$2\lg n \notin \mathbb{N} \rightarrow P\{X > 2\lg n\} = P\{X > \lfloor 2\lg n \rfloor\}$

$\le 2^{-\lfloor 2\lg n \rfloor} < \frac{2}{n^2} = O\left(\frac{1}{n^2}\right)$

Let the random variable X_i denote the number of probes required by the ith insertion. You have shown in part (b) that $\Pr\{X_i > 2\lg n\} = O(1/n^2)$. Let the random variable $X = \max\{X_i : 1 \le i \le n\}$ denote the maximum number of probes required by any of the n insertions.

c. Show that $\Pr\{X > 2\lg n\} = O(1/n)$.

d. Show that the expected length $\mathrm{E}[X]$ of the longest probe sequence is $O(\lg n)$.

c. let A denote $X > 2\lg n$

let A_i denote $X_i > 2\lg n$, $A = \bigcup_{i=1}^{n} A_i$

$$P\{A\} \le \sum_{i=1}^{n} P\{A_i\} \le n\, O\left(\frac{1}{n^2}\right) = O\left(\frac{1}{n}\right)$$

d.

$$E[X] = \sum_{k=1}^{n} k\, P\{X=k\} = \sum_{k=1}^{\lceil 2\lg n\rceil} k \cdot P\{X=k\} + \sum_{k=\lceil 2\lg n\rceil+1}^{n} k \cdot P\{X=k\}$$

$$\le \sum_{k=1}^{\lceil 2\lg n\rceil} \lceil 2\lg n\rceil P\{X=k\} + \sum_{k=\lceil 2\lg n\rceil+1}^{n} n\, P\{X=k\}$$

$$= \lceil 2\lg n\rceil \sum_{k=1}^{\lceil 2\lg n\rceil} P\{X=k\} + n \sum_{k=\lceil 2\lg n\rceil+1}^{n} P\{X=k\}$$

$$\Rightarrow E[X] \le \lceil 2\lg n\rceil + n\, O\left(\frac{1}{n}\right) = O(\lg n) \quad \#$$

11-2 Searching a static set

You are asked to implement a searchable set of n elements in which the keys are numbers. The set is static (no INSERT or DELETE operations), and the only operation required is SEARCH. You are given an arbitrary amount of time to preprocess the n elements so that SEARCH operations run quickly.

a. Show how to implement SEARCH in $O(\lg n)$ worst-case time using no extra storage beyond what is needed to store the elements of the set themselves.

b. Consider implementing the set by open-address hashing on m slots, and assume independent uniform permutation hashing. What is the minimum amount of extra storage $m - n$ required to make the average performance of an unsuccessful SEARCH operation be at least as good as the bound in part (a)? Your answer should be an asymptotic bound on $m - n$ in terms of n.

a. sort and binary search.

b. $p(\text{unsuccessful search}) = \frac{1}{1-\frac{n}{m}}$

$p \le c\lg n \Rightarrow \frac{m}{m-n} \le c\lg n \Rightarrow m \le cm\lg n - cn\lg n$

$m\lfloor c\lg n - 1 \rfloor \ge cn\lg n \Rightarrow m \ge \frac{cn\lg n}{c\lg n - 1}$

$\Rightarrow m - n \ge \frac{cn\lg n - (cn\lg n - n)}{c\lg n - 1} = \frac{n}{c\lg n - 1} = \Omega\left(\frac{n}{\lg n}\right)$