CLRS Exercise

Tongda Xu

October 23, 2018

1   7

1.1   7.3

1.1.1   a

This is certain concerning the *Randomized* procedure, the probability of any index i is chosen from $[0, n-1]$ is:

$Pr(pivot = i) = \frac{1}{n}$

$E(X_i) = 1 * Pr(pivot = i) + 0 * Pr(pivot \neq i) = \frac{1}{n}$

1.1.2   b

It is certain that if ith element is chosen as pivot, $Random - Parition$ cost $\Theta(n)$ time, and it will call $QuickSort[1, q-1], QuickSort[q+1, n]$ recursively. Concerning only the first $Parition$, this would be the result:

$E(T(n)) = \Sigma_{i=1}^{n} Pr(pivot = i)(T(i-1) + T(n-i) + \Theta(n))$

$= \Sigma_{i=1}^{n} X_i(T(i-1) + T(n-i) + \Theta(n))$

1.1.3   c

Concerning $X_i = \frac{1}{n}$

$E(T(n)) = \Sigma_{i=1}^{n} \frac{1}{n}(T(i-1) + T(n-i) + \Theta(n))$

$= \Sigma_{i=1}^{n} \frac{1}{n} T(i-1) + \Sigma_{i=1}^{n} \frac{1}{n} T(n-i) + \Sigma_{i=1}^{n} \frac{1}{n} \Theta(n)$

$= \frac{2}{n} \Sigma_{i=1}^{n-1} T(i) + \Theta(n)$

1.1.4   d

$\Sigma_{k=2}^{n-1} k lg k$

$\leq lg \frac{n}{2} \Sigma_{k=2}^{\frac{n}{2}} k + lgn \Sigma_{k=\frac{n}{2}}^{n-1} k$

$= lgn \Sigma_{k=2}^{n-1} k - lg2 \Sigma_{k=2}^{\frac{n}{2}} k$

$= lgn \frac{(n+1)(n-2)}{2} - \frac{(\frac{n}{2}+2)(\frac{n}{2}-1)}{2}$

$\leq lgn \frac{n^2}{2} - \frac{n^2}{8}$

by Calculus, we have:

$(\frac{1}{2}x^2 lgx - \frac{1}{4}x^2)|_1^{n-1} \leq E(T(n)) \leq (\frac{1}{2}x^2 lgx - \frac{1}{4}x^2)|_2^n$

1.1.5 e

Proof of $E(T(n)) = O(nlgn)$:

Assume that $\forall k \in [1, n-1], \exists c, E(T(k)) \leq cklgk - \Theta(k)$

For $k = n, E(T(n)) \leq \frac{n}{2}c(lgn\frac{n^2}{2} - \frac{n^2}{4} - \Theta(n^2)) + \Theta(n) \leq cnlgn - \Theta(n)$

Proof of $E(T(n)) = \Omega(nlgn)$:

Assume that $\forall k \in [1, n-1], \exists c, E(T(k)) \geq cklgk + \Theta(k)$

For $k = n, E(T(n)) \geq \frac{n}{2}c(lgn\frac{(n-1)^2}{2} - \frac{(n-1)^2}{4} + \Theta(n^2)) + \Theta(n) \geq cnlgn + \Theta(n)$

$\rightarrow E(T(n)) = \Theta(nlgn)$

1.2 7.5

1.2.1 a

From counting Theorem, it could be noticed that:

$p_i = \frac{(i-1)(n-i)}{C_n^3} = \frac{6(i-1)(n-i)}{n(n-1)(n-2)}$

1.2.2 b

$Pr(i = medium)(normal) = \frac{1}{n}$

$Pr(i = medium)(3part) = \frac{6(\frac{1}{2}n-1)(n-\frac{1}{2}n)}{n(n-1)(n-2)} = \frac{3}{2}\frac{1}{n}$

$Pr(3part) - Pr(normal) = \frac{1}{2}\frac{1}{n}$

1.2.3 c

Consider $f_{diff} = \int_{\frac{n}{3}}^{\frac{2}{3}n}(\frac{6(i-1)(n-i)}{n(n-1)(n-2)} - \frac{1}{n})di$

$= \frac{(-2i^3 + 3(n+1)i^2 - 6ni - (n-1)(n-2)i)|_{i=\frac{1}{3}n}^{i=\frac{2}{3}n}}{n(n-1)(n-2)}$

$lim_{n \to \infty} f_{diff} = \frac{4}{27}$

1.2.4 d

Consider we are so lucky that each partition we choose the median:

In the Iteration tree, we have:

$T(n) = \begin{cases} c & n = 1 \\ 2T(\frac{1}{2}n) + n & n > 1 \end{cases}$

The $\Omega(nlgn)$ is kept even in best case.

2 8

2.1 8.1-1

n-1 times, since we need n elements to formulate

2.2   8.1-2

$\Sigma_1^n lgk < \int_1^{n+1} lgkdk = (klgk - k)_1^n = (nlgn - n) - (0 - 1) = nlgn - n + 1$

2.3   8.1-3

$\leftrightarrow$ proof at least half of branch is longer than h
Consider a decision tree with $n!/2$ elements
$\leftrightarrow$ proof at least half of branch is longer than h
Consider a decision tree with $n!/n$ elements
$\leftrightarrow$ proof at least half of branch is longer than h
Consider a decision tree with $n!/2^n$ elements, this is not significant enough and could leave only $\Omega(lg\frac{n!}{2^n}) = \Omega(nlgn - n) = \Omega(nlgn)$ elements

2.4   8.2-4

Consider a trim version of counting sort, build the C map up and query directly:

COUNTING-SORT-TRIM(A, k)

```
1   C[]
2   for i = 0 to k
3       C[i] = 0
4   for j = 1 to A.length
5       C[A[j]] + +
6   for m = 1 to k
7       C[m]+ = C[m - 1]
8   return C[m]
```

DIRECT-QUERT(A, k, a, b)

```
1   C = COUNTING-SORT-TRIM(A, k)
2   if a < 1
3       return C[b]
4   else return C[b] - C[a - 1]
```

2.5   8.3-2

Heapsort is not stable
The scheme would be very similar to counting sort and takes $\Theta(n)$ time

2.6   8.3-4

First, with $O(n)$ time: convert n numbers k_{10} into k_n which has 3 digits.
Second, with $O(d(n+n))$ time ($Lemma 8.3$): Radix sort n 3-digit numbers with each digits take up to n possible values.

DIGITSCONVERT(X)

1 $result[]$
2 **for** $i = 2$ **downto** 0
3      $result[i] = X/n^i$
4      $X = X \bmod n^i$
5 **return** $result$

SORT(A, x)

1 $result[]$
2 **for** each S in A
3      $S = $ DIGITSCONVERT(S)
4 RADIX-SORT(A, x)

3 9

3.1 9.2-1

once $p == r$, the function return and recursion end.

3.2 9.2-2

It is because $\forall k, X_k = \frac{1}{n}$, giving information on which k would not effect observation

3.3 9.2-3

RANDOMIZED-SELECT-ITER(A, p, r, i)

1 **while** 1
2   **if** $i == k$
3   **return** $A[i]$
4   **else**
5          $q = $ RANDOM-PARTITION(A, p, r)
6   **if** $i < k$
7              $r = q - 1$
8   **else** $p = q + 1, i = i - k$

3.4 9.2-4

The worst case is reverse side:
$pivot = 9, 8, 7, 6, 5, 4, 3, 2, 1, 0$

3.5 9.1

3.5.1 a

Sorting: MERGE-SORT(A) in worst case $O(nlgn)$
Query: CALL-BY-RANK(A, k) i times in worst case $O(i)$, here we assume manip-

ulating $O(n)$ space cost $O(n)$ time.

3.5.2 b

Building: BUILD-MAP-HEAP(A) in worst case $O(n)$
Query: calling EXTRA-MAX(A, k) i times in worst case $O(ilgn)$

3.5.3 c

Selecting: SELECT(A, i) in worst case $O(n)$
Sorting: MERGE-SORT(A') in worst case $O(ilgi)$

3.6 9.2

3.6.1 a

$\Sigma_1^{k-1} w_i = \Sigma_1^{k-1} \frac{1}{n} = \frac{k-1}{n} < \frac{1}{2}$
$\Sigma_{k+1}^n = \frac{n-k}{n} \leq \frac{1}{2}$

3.6.2 b

WEIGHT-MEDIAN(A)

```
1   w[] = SORT(A).weight
2   n = w.length
3   for i = 1 to n
4       w[i] = w[i] + w[i − 1]
5   return FIND(w[], 1/2)
```

3.6.3 c

SUM$(w_1, w_i, lasti, lastsum)$

```
1   if i > lasti
2       return lastsum + NORMAL-SUM( w_{lasti,i} )
3   else return lastsum − NORMAL-SUM( w_{i,lasti})
```

WEIGHT-MEDIAN-LINEAR(A)

```
1   while 1
2       if sum[w_1, w_i, lasti, lastsum] < 1/2, sum[w_1, w_{i+1}, lasti, lastsum] > 1/2
3           return i
4       else
5           lastsum = sum[w_1, w_i, lasti, lastsum], lasti = i
6           if sum[w_1, w_i] < 1/2
7               i = MEDIAN(A, i, r)
8           else i = MEDIAN(A, p, i)
```

We will experience *logn* literation, but the load is decreasing logarithmically, so the result is linear. Notice the sum is special here, calculating the difference only.

3.7   9.4

3.7.1   a

$k \le i$ or $k \ge j : 0$
$i < k < j : \frac{2}{j-i+i}$

3.7.2   b

3.7.3   c

3.7.4   d

4   11

4.1   11.1-1

Consider $vector < bool > A, a.size() = m$, just store the bool value of $key = m$ exist or not.

SEARCH(A, key)
1   **if** $A(key)$
2   **return** key
3   **else return** NIL

INSERT(A, key)
1    $A(key) = 1$

DELETE(A, key)
1    $A(key) = 0$

4.2   11.2

4.2.1   a

Consider for a ball i fall into a specific bucket $Pr(i) = \frac{1}{n}$
Then consider Binomial Distribution, $Pr(k) = C_n^k Pr(i)^k (1 - Pr(i))^{n-k}$

4.2.2   b

Consider random picking a slot, the probability of that slot is maximum is $Pr_{max} = \frac{1}{n}$, and it contains k elements Q_k. for conditional probability, we have:
$P_k = Pr_{i=k|max} = \frac{Pr(i=k \cap max)}{Pr_{max}} \le \frac{Pr(i=k)}{Pr_{max}} = nQ_k$

6

4.2.3   c

Proof:

$Q_k = (\frac{1}{n})^k (\frac{n-1}{n})^{n-k} C_n^k$

$= \frac{(n-1)^{n-k}}{n^n} \frac{\Pi_0^{k-1} n-k}{k!}$

$\leq \frac{n^n}{n^n} \frac{1}{k!}$

$= \frac{e^k}{k^k} \frac{1}{k^{\frac{1}{2}} (1+\Theta(\frac{1}{n}))}$

$\leq \frac{e^k}{k^k}$

4.2.4   d

Proof for Q_{k_0}:

$Q_{k_0} = \frac{e^{(\frac{clgn}{lglgn})}}{(\frac{clgn}{lglgn})^{\frac{clgn}{lglgn}}}$

$= \frac{n^{\frac{clg\frac{e}{c}}{lglgn}}}{\frac{n^c}{n^{\frac{clglglgn}{lglgn}}}} = n^{\frac{clg\frac{e}{c}+clglglgn}{lglgn}-c}$

It would not take effort to notice that since $lim_{n\to\infty} \frac{clg\frac{e}{c}+clglglgn}{lglgn} = 0$

$\forall c > 3 + \epsilon, Q_{k_0} = O(\frac{1}{n^3})$

And $P_k \leq nQ_k \to P_k = O(\frac{1}{n^2})$

4.2.5   e

$E(M) = \Sigma_{M=1}^n MPr(M) < nPr(M > \frac{clgn}{lglgn}) + \frac{clgn}{lglgn} Pr(M \leq \frac{clgn}{lglgn})$

A stronger conclusion to note:

$E(M) = \Sigma_{M=1}^n MPr(M) < MPr(M > \frac{clgn}{lglgn}) + \frac{clgn}{lglgn} Pr(M \leq \frac{clgn}{lglgn})$

$\leq \int_{\frac{clgn}{lglgn}}^{\infty} \frac{1}{n} dn + 1 * \frac{clgn}{lglgn}$

$= lg(\frac{clgn}{lglgn}) + \frac{clgn}{lglgn}$

$= O(\frac{clgn}{lglgn})$

5   15

5.1   15.1-1

$2^n - 1 = \Sigma_{j=0}^{n-1} 2^j$

5.2   15.1-2

Do not know how!

5.3   15.1-3

See Code

Ex 15.2.4

2*(5)*(N-5)
2*(4)*(N-4)
2*(3)*(N-3)
2*(2)*(N-2)
2*(1)*(N-1)
0

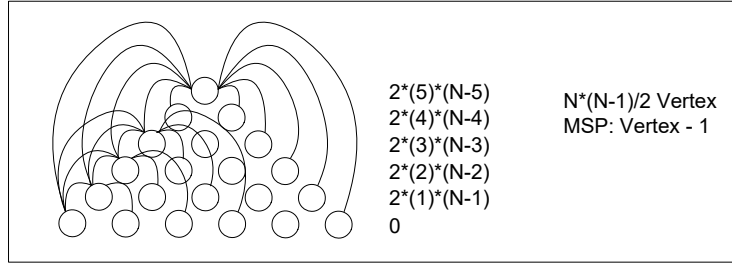N*(N-1)/2 Vertex
MSP: Vertex - 1

Figure 1: 15.2-4

5.4   15.1-4

See Code

5.5   15.1-5

See Code

5.6   15.2-1

See Code

5.7   15.2-2

See Code

5.8   15.2-3

Assume that $\forall k \leq n - 1, T(k) \geq c2^k$
Then $T(n) = \Sigma_{k=1}^{n-1} T(k)T(n-k) = (n-1)c^2 2^n > c2^n$
So $T(n) = \Omega(n), \omega(n)$

5.9   15.2-4

See Figure 1

5.10   15.2-5

For each level $h(i) = i(n-i)$
For tree $T(n) = 2\Sigma_{i=1}^{n-1} i(n-i)$
$= \frac{3n^3 + 3n^2}{3} - \frac{2n^3 + 3n^2 + n}{3}$
$= \frac{n^3 - n}{3}$

8

5.11 15.2-6

Assume that $\forall k \leq n-1, N(k) = k-1$
Then $N(n) = N(n-1) + 1$
So $N(n) = n-1$

5.12 15.3-1

running through: $T(n) = n * P_n^n = n * n! > 4^n$
running recursion: $T(n) = 2\Sigma_{i=1}^{n-1}4^i + n = \frac{8}{3}4^{n-1} + n \leq 4^n$
running through takes longer

5.13 15.3-2

no overlapping subproblem call

5.14 15.3-3

Yes

5.15 15.3-4

Do not know how!

5.16 15.4-1

See code

5.17 15.4-2

See code

5.18 15.4-3

See code

5.19 15.1

$\text{LSP}(s, t, G)$

```
1   r = G, size()
2   DPs[r] = 0
3   DPr[r] = path(s, t)
4   max = -∞
5   for i = 1 to r
6       max(DPs[j] + DPr[r - j] + what)
7   return max
```

5.20   15.1