

Sample Solutions to Homework #1

1. (10)

(a) See Figure 1.

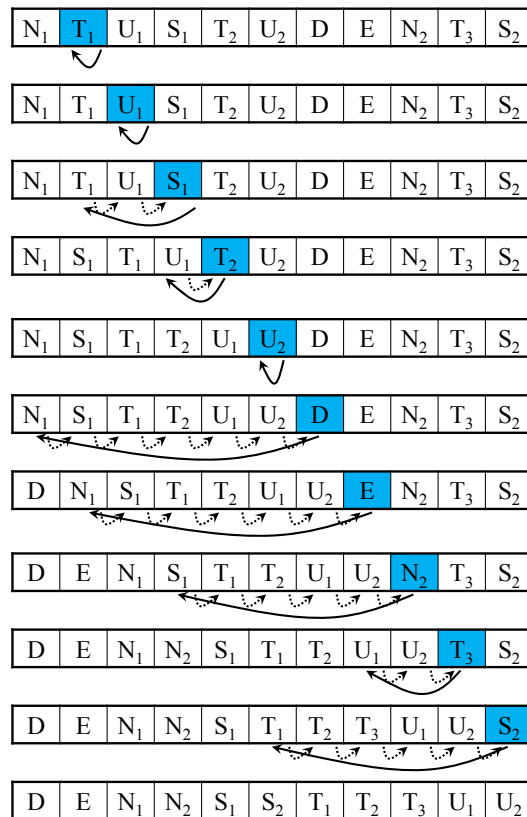


Figure 1: The process steps for Problem 1(a).

(b) See Figure 2.

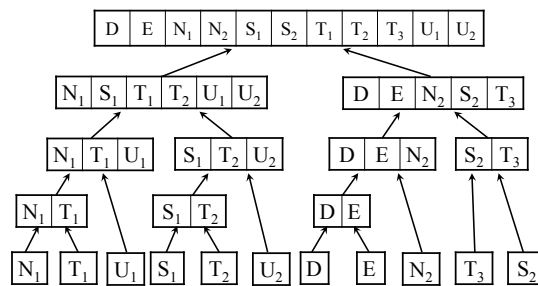


Figure 2: The process steps for Problem 1(b).

2. (20)

- (a) – n : $\text{length}[A]$.
- t_j : # of times that the inequality in line 5 holds at iteration j .
- **Pseudocode:**

SelectionSort(A)	cost	times
1. for $j \leftarrow 1$ to $\text{length}[A] - 1$ do	c_1	n
2. $key \leftarrow A[j]$;	c_2	$n - 1$
3. $key_pos \leftarrow j$;	c_3	$n - 1$
4. for $i \leftarrow j + 1$ to $\text{length}[A]$ do	c_4	$\sum_{j=1}^{n-1} (n - j + 1)$
5. if $A[i] < key$	c_5	$\sum_{j=1}^{n-1} (n - j)$
6. then $key \leftarrow A[i]$;	c_6	$\sum_{j=1}^{n-1} t_j$
7. $key_pos \leftarrow i$;	c_7	$\sum_{j=1}^{n-1} t_j$
8. $A[key_pos] \leftarrow A[j]$;	c_8	$n - 1$
9. $A[j] \leftarrow key$;	c_9	$n - 1$

- (b) **Loop invariant:** After performing the j th iteration of the external **for** loop the subarray $A[1..j]$ contains the j smallest numbers in A , sorted in non-decreasing order.

- **Initialization:** The first iteration finds the smallest number in A and puts it in $A[1]$. Obviously the subarray is sorted, since it contains only one number.
- **Maintenance:** We must show that if the loop invariant was true at the end of iteration j , it will still hold at the end of iteration $j + 1$. Thus, we can assume that at the end of iteration j the subarray $A[1..j]$ contained the j smallest numbers in A sorted in non-decreasing order. The next iteration finds the smallest number in $A[j + 1..n]$ and puts it in $A[j + 1]$. By our assumption, this number is equal to or greater than any of the numbers in $A[1..j]$. Therefore, the new subarray $A[1..j + 1]$ is also sorted in non-decreasing order.
- **Termination:** After the last iteration j equals $n - 1$, the subarray $A[1..j]$ contains the $n - 1$ smallest numbers, sorted in non-decreasing order. By the same argument as before, the entire array must be sorted in non-decreasing order.

- (c) Because every time we select out the smallest number from the subarray $A[j..n]$, the n th smallest number is the last one in the array. Besides, the n th smallest number is also the largest one in the array. Therefore, we do not need to change its place at all, implying that we need to run for only the first $n - 1$ elements.

- (d) – $T(n) = c_1n + c_2(n - 1) + c_3(n - 1) + c_4 \sum_{j=1}^{n-1} (n - j + 1) + c_5 \sum_{j=1}^{n-1} (n - j) + c_6 \sum_{j=1}^{n-1} t_j + c_7 \sum_{j=1}^{n-1} t_j + c_8(n - 1) + c_9(n - 1)$
- **Best case:** If the array is already sorted, all t_j 's are 0.
Quadratic: $T(n) = (\frac{c_4+c_5}{2})n^2 + (c_1+c_2+c_3+c_8+c_9 + \frac{c_4-c_5}{2})n - (c_2+c_3+c_4+c_8+c_9) = \Theta(n^2)$
- **Worst case:** If the array is in reverse sorted order, $t_j = j, \forall j$.
Quadratic: $T(n) = (\frac{c_4+c_5+c_6+c_7}{2})n^2 + (c_1+c_2+c_3+c_8+c_9 + \frac{c_4-c_5-c_6-c_7}{2})n - (c_2+c_3+c_4+c_8+c_9) = \Theta(n^2)$

3. (10)

Use merge sort the sort the n elements in $\Theta(n \lg n)$ time. Then for each element x_1 , use binary search to find if there exists x_2 such that $x_1 + x_2 = x$. The total run time = $\Theta(n \lg n) + n \cdot \Theta(\lg n) = \Theta(n \lg n)$.

4. (20)

- (a) $\Theta(n)$.
- (b) See Figure 3. Since the power function $\text{POW}(x, i)$ runs in $\Theta(i)$ time, the total time complexity is $\Theta(n^2)$. It is worse than the Horner's rule.

```

1.  $y \leftarrow 0.$ 
2. for  $i \leftarrow 0$  to  $n$  do
3.    $y \leftarrow y + a_i \times \text{POW}(x, i)$ 
4. done

```

Figure 3: The pseudocode to naively evaluate a polynomial.

(c) Beginning with $i = n$, we have

$$y_n = a_n$$

and it holds for the invariant. Then, we assume the invariant holds for $i > m$. For $i = m$, we have

$$\begin{aligned}
y_m &= a_{m+1} + x \cdot y_{m+1} \\
&= a_{m+1} + x \cdot \sum_{k=0}^{n-(m+2)} a_{k+m+2} x^k \\
&= a_{m+1} + \sum_{k=0}^{n-(m+2)} a_{k+m+2} x^{k+1} \\
&= a_{m+1} + \sum_{k=1}^{n-(m+1)} a_{k+m+1} x^k \\
&= \sum_{k=0}^{n-(m+1)} a_{k+m+1} x^k
\end{aligned}$$

That means the invariant holds for any $0 \leq i \leq n - 1$. Then it terminates at $i = -1$:

$$y = \sum_{k=0}^n a_k x^k$$

(d) Since we have proven the initialization, maintenance, and termination of the loop invariant, we have also proven the correctness of the given code fragment.

5. (10)

Let $h(n)$ denote $\max\{f(n), g(n)\}$. Since $f(n) \leq h(n)$ and $g(n) \leq h(n)$, we have $f(n) + g(n) \leq 2 \cdot h(n)$ or $h(n) \geq 1/2 \cdot (f(n) + g(n))$ for all $n \geq 1$. Hence, $h(n) = \Omega(f(n) + g(n))$.

Since $f(n)$ and $g(n)$ are asymptotically nonnegative, there exists a constant n_0 such that $f(n) \geq 0$ and $g(n) \geq 0$ for $n \geq n_0$. For $n \geq n_0$, we therefore have $h(n) \leq f(n) + g(n) = 1 \cdot (f(n) + g(n))$. Hence, $h(n) = O(f(n) + g(n))$. We conclude that $h(n) = \Theta(f(n) + g(n))$.

6. (10)

For $n \geq |a|$, we have $0 \leq n + a \leq 2n$ and hence $(n + a)^b \leq (2n)^b = 2^b \cdot n^b$. Therefore, $(n + a)^b = O(n^b)$.

For $n \geq 2 \cdot |a|$, we have $n + a \geq n/2$ and $(n + a)^b \geq \frac{n^b}{2^b}$. Therefore, $(n + a)^b = \Omega(n^b)$. We conclude that $(n + a)^b = \Theta(n^b)$.

7. (15)

$$\begin{aligned}
2^{2^{n+1}} &> 2^n > \left(\frac{3}{2}\right)^n > (\lg n)^{\lg n} = n^{\lg \lg n} > \\
(\lg n)! &> n^3 > n^2 = 4^{\lg n} > \lg(n!) > 2^{\lg n} > 2\sqrt{2^{\lg n}} > \\
\sqrt{\lg n} &> \ln \ln n > 2^{\lg^* n} > \lg^* n = \lg^*(\lg n) > \lg \lg^* n > n^{\frac{1}{\lg n}} = 1
\end{aligned}$$

8. (10)

Let $x = \frac{1}{\alpha}$ and $y = \frac{1}{1-\alpha}$. Construct the recurrence tree as Figure 4. We can obviously see that $T(n) = \Theta(n \lg n)$.

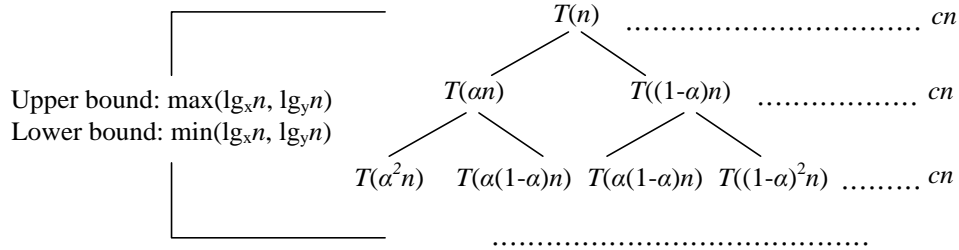


Figure 4: The recursion tree for Problem 8.

9. (20)

- (b) Apply the master theorem: $n^{\log_b a} = n^0$, $f(n) = n^1$, and we can find some $\epsilon > 0$ such that $f(n) = \Omega(n^{\log_b a - \epsilon})$. We also can find that $f(\frac{n}{10}) < cf(n)$ when $\frac{9}{10} < c < 1$. So $T(n) = \Theta(n)$.
- (e) $T(n) = 7T(n/2) + n^2 = \Theta(n^{\lg 7})$. This is a divide-and-conquer recurrence with $a = 7$, $b = 2$, $f(n) = n^2$, and $n^{\log_b a} = n^{\log_2 7}$. Since $2 < \lg 7 < 3$, we have that $n^2 = O(n^{\log_2 7 - \epsilon})$ for some const $\epsilon > 0$. Thus, case1 of the master theorem applies, and $T(n) = \Theta(n^{\lg 7})$.
- (f) Apply the master theorem: $n^{\log_b a} = n^{\frac{1}{2}}$, $f(n) = n^{\frac{1}{2}}$, and thus $T(n) = \Theta(n^{\frac{1}{2}} \lg n)$.
- (h) $T(n) = T(\sqrt{n}) + 1 = \Theta(\lg \lg n)$. The easy way to do this is with a change of variables. Let $m = \lg n$ and $S(m) = T(2^m)$. $T(2^m) = T(2^{m/2}) + 1$, so $S(m) = S(m/2) + 1$. Using the master theorem, $n^{\log_b a} = n^{\log_2 1} = n^0 = 1$ and $f(n) = 1$. Since $1 = \Theta(1)$, case2 of the master theorem applies, and $S(m) = \Theta(\lg m)$. Therefore, $T(n) = \Theta(\lg \lg n)$.

10. (20)

- (b) Solve by recurrence: assuming $n = 5^k$, and using harmonic series (A.7)

$$\begin{aligned}
 T(n) &= 5T\left(\frac{n}{5}\right) + \frac{n}{\lg n} \\
 &= 25T\left(\frac{n}{25}\right) + \frac{n}{\lg n} + \frac{n}{\lg \frac{n}{5}} \\
 &= 125T\left(\frac{n}{125}\right) + \frac{n}{\lg n} + \frac{n}{\lg \frac{n}{5}} + \frac{n}{\lg \frac{n}{25}} \\
 &= 5^k T\left(\frac{n}{5^k}\right) + \frac{n}{\lg n} + \frac{n}{\lg \frac{n}{5}} + \frac{n}{\lg \frac{n}{5^2}} + \dots + \frac{n}{\lg \frac{n}{5^{k-1}}} \\
 &= nT(1) + \frac{n}{\lg 5} + \frac{n}{\lg 25} + \dots + \frac{n}{\lg 5^k} \\
 &= nT(1) + (n \lg 5) \left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{k} \right) \\
 &= cn + (\lg 5)n(\lg(\lg n) + O(1)) \\
 &= \Theta(n \lg \lg n)
 \end{aligned}$$

- (d) The function $T(n)$ grows at least as fast as $T'(n)$ defined recursively by $T'(n) = 3T'(\frac{n}{3}) + \frac{n}{2}$. By master theorem, $T'(n) = \Theta(n \lg n)$ and hence $T(n) = \Omega(n \lg n)$.

To prove $T(n) = O(n \lg n)$, we show that $T(n) \leq c \cdot (n \lg n)$ by induction on n for a suitable constant $c > 0$.

For $n \geq 128$, we have

$$n/3 + 5 < n/2 < n$$

and

$$15 \cdot c \cdot \lg n - n(c - 1/2) < 0.$$

Hence, the induction assumption implies

$$\begin{aligned} T(n) &= 3T(n/3 + 5) + n/2 \\ &\leq 3 \cdot c \cdot (n/3 + 5) \cdot \lg(n/3 + 5) + n/2 \\ &= c \cdot (n + 15) \cdot \lg(n/3 + 5) + n/2 \\ &= c \cdot n \cdot \lg(n/3 + 5) + 15 \cdot c \cdot \lg(n/3 + 5) + n/2 \\ &< c \cdot n \cdot \lg(n/2) + 15 \cdot c \cdot \lg(n/2) + n/2 \\ &< c \cdot n \cdot \lg(n/2) + 15 \cdot c \cdot \lg n + n/2 \\ &= c \cdot n \cdot \lg n + 15 \cdot c \cdot \lg n - n(c - 1/2) \\ &< c \cdot n \cdot \lg n. \end{aligned}$$

(f) Drawing the recursion tree, we get

$$\begin{aligned} T(n) &= n + \frac{7}{8}n + \frac{49}{64}n + \dots + \left(\frac{7}{8}\right)^{\lg n} n \\ &= \Theta(n). \end{aligned}$$

(j) See Figure 5. The depth is $\lg \lg n$, and thus we get $\Theta(n \lg \lg n)$.

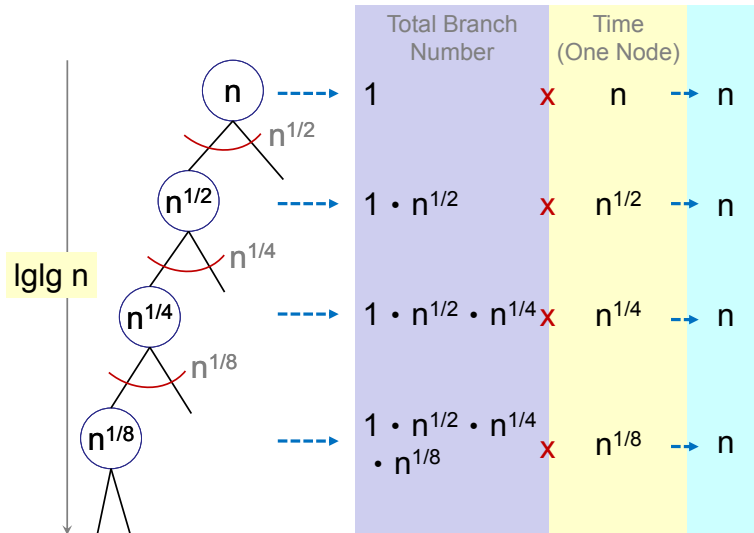


Figure 5: The process steps for Problem 10(j).

11. (30) DIY.