CSCE-629 Homework 2

Wang, Han-Yi
wanghy917@email.tamu.edu

August 28, 2020

Problem 2-2

(a) In addition to provie $A'[1] \leq A'[2] \leq \cdots \leq A'[n]$, we also need to prove that A' contains the same elements as in A.

(b) Loop invariant for the inner loop (lines 2–4): At the start of each iteration, the position of the smallest element of $A[i..n]$ is at most at j.

- Initialization: When $j = n$, it is trivial that the smallest element is at most at n.
- Maintenance: In each iteration,
  - If the smallest item is $A[j]$, it will exchange with $A[j-1]$, so the smallest item is at most at $j-1$ after the iteration, where $j-1 = i$.
  - If the smallest item is not $A[j]$, its index is already less than or equal to $j-1$, where $j-1 \geq i$.
- Termination: The inner loop (lines 2–4) terminates when j is equal to i. Before it terminate, $A[i+1]$ will exchange with $A[i]$ if $A[i+1] < A[i]$ in the last iteration. Therefore, $A[i]$ will be the smallest item in $A[i..n]$ when the algorithm terminate.

(c) Loop invariant for the outer loop (lines 1–4): At the start of each iteration, the subarray $A[1..i-1]$ contains the 1_{st} to $(i-1)_{th}$ smallest elements of A in sorted order.

- Initialization: Before the first iteration, $i = 1$, $A[1..0]$ is empty, so it is sorted.
- Maintenance: From **(b)**, when the inner loop terminates, the smallest item in $A[i..n]$ is in position i. Also the $(i-1)_{th}$ smallest elements of A are already in $A[1...i-1]$, so $A[i]$ is the i_{th} smallest element of A. Therefore $A[1...i]$ is sorted and it contains the 1_{st} to i_{th} smallest elements of A.
- Termination: The outer loop (line 1-4) terminates when i is equal to n. After the last iteration, the subarray $A[1..n-1]$ contains the 1_{st} to $(n-1)_{th}$ elements originally in A, so the n_{th} element is at the n_{th} location, and A is sorted.

(d) The outer loop runs $n-1$ iterations, and the inner loop runs $n-i$ iterations in the i_{th} iteration of outer loop with constant execution time. Therefore, the worst case is:

$$T(n) = \sum_{i=1}^{n-1}(n-i) = \frac{(n-1) \times (n-1)}{2} = \Theta(n^2) \tag{1}$$

, where the worst-case running time is the same as the insertion sort.

Exercise 4.3-1

We guess $T(n) \leq cn^2$, and we have

$$\begin{aligned} T(n) &\leq c(n-1)^2 + n \\ &= cn^2 - 2cn + c + n \\ &= cn^2 + n(1-2c) + c \\ &\leq cn^2 + 1(1-2c) + c \\ &= cn^2 + 1 - c \\ &\leq cn^2 \end{aligned} \tag{2}$$

, where the 4_{th} step holds for $c \geq \frac{1}{2}$ and $n \geq 1$, and the last step holds for $c \geq 1$.

Exercise 4.3-2

We guess $T(n) \leq clg(n-2)$, and we have

$$\begin{aligned} T(n) &\leq c\lg(\lceil n/2 \rceil - 2) + 1 \\ &\leq c\lg(n/2 + 1 - 2) + 1 \\ &= c\lg((n-2)/2) + 1 \\ &= c\lg(n-2) - c\lg 2 + 1 \\ &\leq c\lg(n-2) \end{aligned} \tag{3}$$

, where the last step holds for $c \geq 1$.

Exercise 4.3-8

First, we guess $T(n) \leq cn^2$, and we have

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &\leq 4\left(c(n/2)^2\right) + n \\ &= cn^2 + n \end{aligned} \tag{4}$$

, where we can not find a constant c to make the last step holds. We then make another guess that $T(n) \leq cn^2 - n$, and we have

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &\leq 4\left(c(n/2)^2 - n/2\right) + n \\ &= cn^2 - 2cn + n \\ &\leq cn^2 \end{aligned} \tag{5}$$

, where the last step holds for $c \geq \frac{1}{2}$.

Exercise 4.5-1

$a = 2, b = 4, log_b a = \frac{1}{2}$

(a)
- $\frac{1}{2} > 0$
- $T(n) = \Theta\left(n^{\log_4 2}\right) = \Theta(\sqrt{n})$

(b)
- $\frac{1}{2} = \frac{1}{2}$
- $T(n) = \Theta\left(n^{\log_4 2} lg(n)\right) = \Theta(\sqrt{n} lg(n))$

(c)
- $\frac{1}{2} < 1$
- $T(n) = \Theta(n)$

(d)
- $\frac{1}{2} < 2$
- $T(n) = \Theta(n^2)$

Exercise 15.1-2

i	0	1	2	3	4
p[i]	0	1	5	8	9
d[i]	0	1	2.50	2.67	2.25
r[i] (DP)	0	1	5	8	10
r[i] (Greedy)	0	1	5	8	8+1=9

Given price and length in the above table. The density of each length are $d[1] = 1, d[2] = 2.50, d[3] = 2.67$ and $d[4] = 2.25$. When $n = 4$, the greedy algorithm will first add the price of length 3, which has the highest density, then, it can only add the piece of length 1. As a result, the revenue $r[4] = p[3] + p[1] = 9$. However, the highest revenue is $5 + 5 = 10$ when we applied dynamic programming, so the result of greedy algorithm is incorrect.