20407 Data Structures and Introduction to Algorithms

Maman 11

Robert S. Barnes

Question 1

A) Assume n is even. The given list, $\frac{n}{2}, \frac{n}{2}+1, \ldots, n, 1, 2, \ldots, \frac{n}{2}-1$ is made of 2 ordered lists, the first containing $\frac{n}{2}+1$ numbers, and the second containing $\frac{n}{2}-1$ numbers. Thus there are $(\frac{n}{2}+1)(\frac{n}{2}-1)$ inversions in the list.

Compares: On the first $\frac{n}{2}$ iterations, only one compare is made since the elements are already in order. On the $\frac{n}{2}+1$ iteration we encounter 1. The inner loop executes $\frac{n}{2}+1$ compares as the first $\frac{n}{2}+1$ element are shifted up one place. Then the loop is stopped by the $i \neq 0$ condition. At the end of the outer loop, $\frac{n}{2}+1$ inversions have been removed. On each subsequent outer loop, the inner loop executes $\frac{n}{2}+2$ comparisons, since it is now stopped by the `A[i]>key` condition. Thus we have:

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

$$= \frac{n}{2} + \frac{n^2}{4} + \frac{n}{2} - 2 - 1 = \frac{1}{4}(n^2 + 4n - 12)\ \forall \text{ even } n \geq 4$$

The equation doesn't hold for $n = 2$ since the array is already sorted for that value.

The number of key compares is $\Theta(n^2)$ $\forall$ even $n \geq 4$ and we see that the following hold:

$$\frac{n^2}{4} + n - 3 \leq \frac{n^2}{4} + n \leq n^2 + n^2 \Rightarrow c_2 = 2$$

$$\frac{n^2}{4} + n - 3 \geq \frac{n^2}{4} - 3 \geq \frac{n^2}{16} \Rightarrow c_1 = \frac{1}{4}$$

Copies: The algorithm executes 2 copies for each iteration of the outter loop, regardless of order which gives us $2(n-1)$ copies. Then the inner loop effectively removes one inversion per copy per iteration thus:

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

$$= 2n - 2 + \frac{n^2}{4} - 1 = \frac{1}{4}(n^2 + 8n - 12)\ \forall \text{ even } n \geq 2$$

The number of key copies is $\Theta(n^2)$ $\forall$ even $n \geq 2$ and we see that the following hold:

$$\frac{n^2}{4} + 2n - 3 \leq \frac{n^2}{4} + 2n \leq n^2 + 2n^2 \Rightarrow c_2 = 3$$

$$\frac{n^2}{4} + 2n - 3 \geq \frac{n^2}{4} \geq \frac{1}{4}n^2 \Rightarrow c_1 = \frac{1}{4}$$

B) Assume n is even. We see that for each value of n, there are $\Sigma_{i=0}^{\frac{n}{2}-1}(2i)$ inversions in the list.

Compares: On each iteration of the outer loop, $2i$ inversions are removed resulting in the same number of compares, plus one additional compare to stop the loop. So we have:

$$\Sigma_{i=0}^{\frac{n}{2}-1}(2i+1) = \Sigma_{i=0}^{\frac{n}{2}-1}(2i) + \Sigma_{i=0}^{\frac{n}{2}-1}(1)$$

$$= 2\left[\Sigma_{i=0}^{\frac{n}{2}-1}(i)\right] + \Sigma_{i=1}^{\frac{n}{2}}(i)$$

$$= 2\left[\Sigma_{i=0}^{\frac{n}{2}}(i) - \frac{n}{2}\right] + \Sigma_{i=1}^{\frac{n}{2}}(i)$$

$$= 2\left[\frac{(n/2)^2 + n/2)}{2} - \frac{n}{2}\right] + \frac{n}{2}$$

$$= \frac{n^2}{4} + \frac{n}{2} - n + \frac{n}{2} = \frac{n^2}{4}$$

The number of compares in $\Theta(n^2)$ $\forall$ even $n \geq 1$ and we see that the following holds:

$$\frac{1}{4}n^2 \leq \frac{n^2}{4} \leq n^2$$

Copies: The routine performs one copy per inversion removed to shift elements up, plus 2 copies for the key on every loop regardless and thus we get:

$$\Sigma_{i=0}^{\frac{n}{2}-1}(2i) + 2(n-1) = \frac{n^2}{4} + \frac{n}{2} - n + 2n - 2$$

$$= \frac{n^2}{4} + \frac{3n}{2} - 2 = \frac{1}{4}(n^2 + 6n - 8)$$

The number of copies is $\Theta(n^2)\ \forall\ n \geq 2$ and we see that the following holds:

$$n^2 \leq \frac{1}{4}(n^2 + 6n - 8) \leq \frac{1}{4}(n^2 + 6n) \leq (n^2 + 6n^2) \leq 7n^2$$

Question 2

A) We'll show that if $f_i(n) = \Theta(g_i(n))$, then:

$$\Sigma_{i=1}^{k}(f_i(n)) = \Theta(\Sigma_{i=1}^{k}(g_i(n))) = \Theta(\max_{1\leq i\leq k}\{g_i(n)\})$$

Assume that for $1 \leq i \leq k$ we have $f_i(n) = \Theta(g_i(n))$ and that all $g_i(n)$ are asymptoticlly positive, since otherwise the g_i sets are empty and the claim is trivially true. This implies that:

$$\exists c_1^i, c_2^i : \forall 1 \leq i \leq k : 0 < c_1^i g_i(n) \leq f_i(n) \leq c_2^i g_i(n)\ \forall\ n \geq n_0^i$$

$$\Rightarrow 0 < \Sigma_{i=1}^{k}(c_1^i g_i(n)) \leq \Sigma_{i=1}^{k}(f_i(n)) \leq \Sigma_{i=1}^{k}(c_2^i g_i(n))\ \forall\ n \geq \max(n_0^i)$$

$$\Rightarrow 0 < \min_{1\leq i\leq k}(c_1^i)\Sigma_{i=1}^{k}(g_i(n)) \leq \Sigma_{i=1}^{k}(f_i(n)) \leq \max_{1\leq i\leq k}(c_2^i)\Sigma_{i=1}^{k}(g_i(n))\ \forall\ n \geq \max(n_0^i)$$

$$\Rightarrow \Sigma(f_i(n)) = \Theta(\Sigma(g_i(n))$$

For simplicty assume $g_m(n) = \max_{1\leq i\leq k}\{g_i(n)\}$. We look for $c_1, c_2 > 0$ such that:

$$0 < c_1 g_m(n) \leq \Sigma_{i=1}^{k}(g_i(n)) \leq c_2 g_m(n)\ \forall n \geq \max(n_0^i)$$

By assumption, all the g_i are asymptotically positive, so we can divide by g_m:

$$0 < c_1 \leq \frac{\Sigma_{i=1}^{k}(g_i(n))}{g_m(n)} \leq c_2 \forall n \geq \max(n_0^i)$$

The first part of the inequality holds since each g_i is asymptotically positive. The second part holds because $g_m(n) \geq g_i(n)\ \forall i$ which implies that for each term in the summation $\frac{g_i(n)}{g_m(n)} \leq 1$ holds. In fact, since $g_m(n)$ equals some $g_i(n)$ at each point we have that:

$$1 < \frac{\Sigma_{i=1}^{k}(g_i(n))}{g_m(n)} \leq k$$

and thus:

$$0 < g_m(n) \leq \Sigma_{i=1}^{k}(g_i(n)) \leq kg_m(n) \; \forall \; n \geq \max(n_0^i)$$

$$\Rightarrow \Sigma(g_i(n)) = \Theta(max_{1\leq i\leq k}\{g_i(n)\})$$

$$\Rightarrow \Sigma(f_i(n)) = \Theta(max_{1\leq i\leq k}\{g_i(n)\})$$

by transitivity.

B) Assume: $f_1(n) = O(g_1(n)) \wedge f_2(n) = \Theta(g_2(n))$

$$\Rightarrow \exists n_0^1, c_1^1 : f_1(n) \leq c_1^1 g_1(n) \; \forall \; n \geq n_0^1$$

$$\exists n_0^2, c_1^2, c_1^2 : 0 < c_1^2 g_2(n) \leq f_2(n) \leq c_2^2 g_2(n) \; \forall \; n \geq n_0^2$$

$$\Rightarrow f_1(n) + f_2(n) \leq c_1^1 g_1(n) + c_2^2 g_2(n) \; \forall \; n \geq max(n_0^1, n_0^2)$$

$$\Rightarrow f_1(n) + f_2(n) \leq max(c_1^1, c_2^2)\,[g_1(n) + g_2(n)] \;\; \forall \; n \geq max(n_0^1, n_0^2)$$

$$\Rightarrow f_1(n) + f_2(n) = \Theta(g_1(n) + g_2(n))$$

C) Assume $f_1(n) = \Theta(g_1(n)) \wedge f_2(n) = \Omega(g_2(n))$

$$\Rightarrow \exists n_0^1, c_1^1, c_1^1 : 0 < c_1^1 g_1(n) \leq f_1(n) \leq c_2^1 g_1(n) \; \forall \; n \geq n_0^1$$

$$\exists n_0^2, c_1^2 : 0 < c_1^2 g_2(n) \leq f_2(n) \; \forall \; n \geq n_0$$

$$\Rightarrow 0 < c_1^1 g_1(n) + c_1^2 g_2(n) \leq f_1(n) + f_2(n) \; \forall \; n \geq max(n_0^1, n_0^2)$$

$$\Rightarrow 0 < min(c_1^1, c_1^2)[g_1(n) + g_2(n)] \leq f_1(n) + f_2(n) \; \forall \; n \geq max(n_0^1, n_0^2)$$

$$\Rightarrow f_1(n) + f_2(n) = \Omega(g_1(n) + g_2(n))$$

Question 3

A) By definition, `r(A[i])` equals the total number of elements in `A` which are less than `A[i]`, plus the number of elements to the left of `A[i]` which are equal to `A[i]`, plus one. But this is nothing more than the index of `A[i]` in `A` after performing a stable sort. Since the indexes of each element in `A` after performing a stable sort are simply $< 1, \ldots, n >$, then `R` must be a permutation of these numbers.

B) For an array of length 1, `Rank(A,R)` produces the correct output since `A[1] <= A[1]` and we get `R[1]=1`.

Assume that for $j = n - 1$ the two loops produce the correct output. On the next outter loop $j = n$ and the inner loop advance through `A` comparing each element to `A[n]`. Before each iteration of the inner loop starts, `R[i]` contains the rank of `A[i]` in the array `A[1..n-1]` for $i < n$. For $i < n$, if `A[i]>A[n]`, then `R[i]` is incremented and has it's final correct value. Otherwise, if `A[i]<=A[n]`, then `R[n]` is incremented. This happens once for each value in `A[1..n-1]` which is less than or equal to `A[n]` and then once more when `j=n` since `A[n]<=A[n]`. Thus, at the end of the loop `R[n]` contains the rank of `A[n]`, and `R[1..n-1]` contain the corresponding ranks of `A[1..n-1]`.

C) `R[i]` contains the position of `A[i]` in `A` after performing a stable sort. Thus `U[R[i]] = A[i]` copies `A[i]` to it's correct position in `A` and then the second loop copies the sorted array `U` back into `A`.

D) In the call to `RANK`, the loops always execute

$$\Sigma_{j=1}^{n} \Sigma_{i=1}^{j} (1) = \Sigma_{j=1}^{n} (j) = \frac{n(n+1)}{2}$$

compares in all cases. In `RANK-SORT`, all the elements of `A` are copied to `U`, and then back to `A`, resulting in $2n$ copies in all cases.

E) Since each `R[i]` contains the corresponding index of `A[i]` after performing a stable sort, then `A` is sorted if and only if `R` is sorted, since `RANK-SORT1` performs identical swaps on the two arrays.

For an array of length one we have `R[1]=1`, `i=1` and the function does nothing, `R` and `A` are sorted.

Assume that `R[1..k]` is sorted for some $k < n$. Then `R[k+1..n]` contains some permutation of the numbers $< k+1 \ldots n >$. If `R[i]=k+1` then we're done and now `R[1..k+1]` is sorted. If not, then the value at position `i` is swapped with the value at position `R[i]`, which is the correct position of the value at `i`. Since each iteration of the loop places at lesat one value in `R[k+1...n]` into the correct position, then the loop terminates after a maximum of $n-k+1$ iterations. At that point the correct value will be in `R[k+1]` meaning that `R[1..k+1]` and `A[1..k+1]` are now sorted.

F)

Worst Case:

Compares: Since `RANK-SORT1` calls `RANK`, then it always performs exactly $\frac{n(n+1)}{2}$ compares on A. Copies: 3(n-1) Because the last swap always puts 2 elements in their correct place and each previous swap places one element in it's correct position.

Best Case:

Cmps $= \frac{n(n+1)}{2}$ Copies $= 0$

G)

4 3 1 2

2 3 1 4

3 2 1 4

1 2 3 4

H) Since `RANK` always takes $(n^2+n)/2$ compares to complete, then both algorithms worst case run time is $\Theta(n^2)$. The space complexity is $\Theta(n)$ in both cases since both algorithms allocate `R` to pass to `RANK`.

I) Since the `RANK(A,R)` function always runs in $\Theta(n^2)$ time, then the best, worst, and average case times for both sorting algorithms is $\Theta(n^2)$.

Question 4

A) Show that: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Leftrightarrow f(n) = o(g(n))$

1) Assume that $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ and that $c > 0 \in \mathbb{R}$. We look for some value of n_0 such that:

$$0 \leq f(n) < cg(n) \; \forall n \geq n_0$$

The existance of the limit implies that $g(n)$ is asymptotically positive, thus we can divide by $g(n)$ and thus we are looking for some n_0 such that

$$0 \leq \frac{f(n)}{g(n)} < c \; \forall \; n \geq n_0$$

The existance of the limit implies that for any constant value of c which we choose, no matter how close to 0, we can always choose a value of n such that $\frac{f(n)}{g(n)}$ is smaller and the above inequality will hold.

2) Assume that $\forall \; c > 0 \in \mathbb{R} \; \exists n_0 : 0 \leq \frac{f(n)}{g(n)} < c \; \forall \; n \geq n_0$. $0 < cg(n)$ implies that $g(n)$ is asymptotically positive and that we can divide by $g(n)$. Thus no matter how close to 0 we choose c, we can always choose a value for n_0 such that $0 \leq \frac{f(n)}{g(n)} < c \; \forall \; n \geq n_0$. This implies $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

B,C) We have that

$$\lim_{n \to \infty} \frac{n^k \lg n}{n^{k+\epsilon}} = \lim_{n \to \infty} \frac{\lg n}{n^\epsilon} = 0 \; \forall \; \epsilon > 0, k$$

Thus $n^k \lg n = o(n^{k+\epsilon})$ and $n^k \lg n$ is a lower order term, implying that the following holds:

$$1 \times n^{k+\epsilon} \leq n^{k+\epsilon} + n^k \lg n \leq 2n^{k+\epsilon}$$

$$\Rightarrow n^{k+\epsilon} + n^k \lg n = \Theta(n^{k+\epsilon})$$