CSE220: Midterm Exam Solution

Instuctor: Saswati Sarkar

Problem 1 (6pts)

There can be many heaps from the given inputs.

Problem 2 (4pts)

$n^2 = O(n^3 - \sqrt{n})$ is True

$\frac{1}{n} = O(\log n)$ is True

$n^3 = o(n^2 + n\log n)$ is False

$\sqrt{n}\log n = o(n)$ is True

Problem 3 (10pts)

Design:

1. Divide input array into two part at half($\left\lceil \frac{left+right}{2} \right\rceil$), while left $<$ right.

2. Max of both halves is the max of the max of he two halves. This is the conquer.

Complexity:

The recurrence for the running time $T(n)$ of this algorithm is:

$$T(n) = 2T(\frac{n}{2}) + c$$

Using Master's Theorem, we conclude that $T(n)$ is $O(n)$.

Problem 4 (10pts)

Insertion sort makes a single comparison for each element for presorted input. Thus, if we have n of input, we need O(n) running time.

For quick sort, the running time depends on how we choose the pivot number.

If we choose the middle one of the first, last and the center element, we could always divide the given input by half because it is already sorted. So the recurrence is

$$T(n) = 2T(\frac{n}{2}) + cn$$

Using Master's Theorem, we conclude that $T(n)$ is $O(n\log n)$.

Next, if we choose first element of input as a pivot value, we will always choose the lowest because it is already sorted. So partition will always be unbalanced for every step. The recurrence is

$$T(n) = 2T(n-1) + cn, \;\; n > 1$$

Using Master's Theorem, we conclude that $T(n)$ is $O(n^2)$.

Problem 5 (10pts)
Design:

Let's define additional array T whose size is at least the maximum value of list S. (Actually, we don't know the maximum value of S. But because we don't have any limitation on storage, we could assume T has a infinite index.) Then the values in array T are decided by the following rules:

1. The value of all elements in T is intially 0.
2. The value in T is 1 only for the element whose index is S[k]$(0 \leq k \leq n-1)$. Shortly, T[S[k]] = 1.

Now we could find out whether there exist two elements in S whose sum equals x by the following procedure.

```
FindIt
  for(k = 0; k ¡ n; k++)
  {
    index = x-S[k];
    if((index ≥ 0) and (T[index] == 1))s
      return true;
  }
  return false;
```

Complexity:
From the algorithm above, we could know that the worst case is when there is no such two elements in S whose sum equals x. For that case we should check all of the elements in S. Since there is n integers in S, the complexity is O(n) in worst case.

Problem 6 (10pts)
Let's say that DeleteMin(S), push(S), pop(S) each take O(1). Then if you push n elements in the stack and subsequently do DeleteMin n times, you will have a sorted output in O(n) since you will do n times $O(1)$ operation. But we know from the following theorem(p249 in text book) that lower bound is $\Omega(n \log n)$ for sorting using comparison.

Theorem 7.7: Any sorting algorithm that uses only comparisons between elements requires $\Omega(n \log n)$ comparisons.

So we do not believe the Professor.