Exam 3

Yaokun Wu

1.

```
Sort(A[1...n]):
    if n=1:
        return.
    else:
        LIdx ← 1
        maxval ← A[LIdx]
        for i ← 2 to n
            if A[i] > maxval:
                maxval = A[i]
                LIdx = i
        if i ≠ n:
            Reverse(LIdx)
            Reverse(n)
        Sort(A[1...n-1]).
```

Time complexity: Each time Reverse operation performed at most twice.

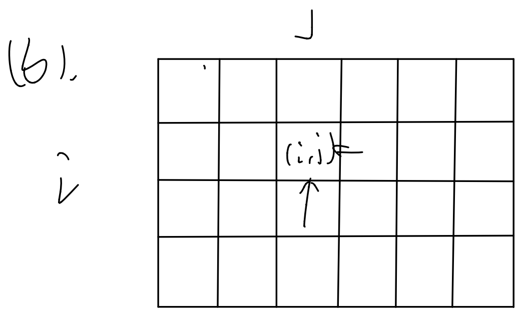
$\Rightarrow$ $O(n)$ Reverse opertion to sort entire array.

2.

(a).

$$BestScore(i,j) = \begin{cases} A[i,j] & \text{if } i=n \text{ and } j=n. \\ \max(BestScore(i+1,j), BestScore(i,j+1)) + A[i,j] & \text{otherwise.} \end{cases}$$

(b).



Based on the recursive definition

We can compute the BestScore in the

order that i from n to 1

and j from n to 1.

After calculating BestScore starting at each grid.

We can traverse the grid to find $\max_{\substack{i=1 \text{ to } n \\ j=1 \text{ to } n}}(BestScore(i,j))$

And return the maximum score.

Time complexity:

$O(n^2)$ because for each grid it takes constant time to compute. and there are n^2 grids.

/

3.

Let $G' = (V', E')$ where V' represent vertices that having k number of edges from source where $k \leq 3|V|$ at vertices V.

Then we will run dijkstra algorithm such that when reaching vertex t with k number of edges when $k \% 3 = 0$, we will return the distance of t from s.

The time complexity is $O(E \log V)$.

4. Contruct a graph G with four types of vertices.

1. Source vertex s'
2. a vertex s_i for each student i.
3. a vertex t_j for each time slot j
4. a ~~target~~ vertex t'.

There are three types of edges.

1. an edge $s' \to s\_i$ with capacity $E(i)$ for each student i.
2. an edge $s\_i \to t\_j$ with capacity 1 for timeslot j if $A[i,j]$ is true.
3. an edge $t\_j \to t'$ with capacity of 2000 for each j.

I want to calculate the maximum flow f^* from s' to t'

and compare it with $\sum_{i=1}^{n} E(i)$.

If $f^* = \sum_{i=1}^{n} E(i)$, then the assignment is possible.

If $f^* < \sum_{i=1}^{n} E(i)$, then there is no way for such assignment.

Time complexity:

Graph G has $O(n+t)$ vertices and at most $O(nt)$ edges. Since Capacity $|f^*| \leq \sum_{i=1}^{n} E(i)$. We can use Ford-Fulkerson for $O(nt \cdot \sum_{i=1}^{n} E(i))$ running time.

5.

(a).

$$MinVCNo(v) = \underline{\quad 0 \quad} + \sum_{w \downarrow v} \underline{MinVCYes(w)}$$

$$MinVCYes(v) = \underline{\quad 1 \quad} + \sum_{w \downarrow v} \min\{MinVCNo(w), \underline{MinVCYes(w)}\}$$

(b). We can evaluate each subproblem in postorder.

The final return value should be

min (MinVCNo(r) , MinVCYes(r)).

(c). Because in (b) we already assume that the graph G is a tree. And MIN VERTEX COVER in class doesn't have this assumption and not every graph can be reduced to tree.

6

(a).

I show that the problem is NP-hard by a reduction from 3COLOR.

Given an arbitrary graph $G=(V,E)$.

Let the vertices represent guests.

If there is an edge between two vertices. We say that the two guests doesn't know each other ahead of time.

I claim that the grouping is possible iff It is possible to solve the 3color problem.

The reduction requires $O(E)$ time because we only need to process each edge exactly once.

(b).

I show that the problem is NP-hard by a reduction from 3SAT.

Let Φ be a 3SAT formula with m varibles and n clauses.

We covert the formula into all requirements problem in following way.

① Let m varibles represent m shapes of balloon and n requirements.

② If varible x_i is in jth clause, then we say that requirement j contains a balloon of shape x_i with color green.

③ If varible $\bar{x}_i$ is in jth clause, then we say that requirement j contains a balloon of shape x_i with color orange.

I claim that the purchase is possible iff the 3SAT problem is satisfiable.

The reduction requires $O(N)$ time where N is the total number of pairs (shape, color) in all requirements because we only need to process each literal exactly once.