# UCLA
# Computer Science Department

## CS180– Midterm
## Algorithms & Complexity

10/30/2018

Name: Yining Wang  1G

UID: 504983099

---

This exam contains 7 pages (including this cover page) and 6 questions.

- Writing has to be legible.
- Express algorithms in bullet form, step by step.

**Distribution of Marks**

| Question | Points | Score |
|---|---|---|
| 1 | 20 | 20 |
| 2 | 20 | 16 |
| 3 | 20 | 13 |
| 4 | 10 | 10 |
| 5 | 20 | 0 |
| 6 | 10 | 6 |
| Total: | 100 | 65 |

1. (20 points) Consider a set of intervals $I_1, I_2, \cdots, I_n$:
   (a) Design a linear time algorithm (assume that intervals are sorted in any manner you wish) that assigns the intervals to the minimum number of processors.
   (b) Prove the correctness of your algorithm.

(a) assume the intervals are sorted by their start ~~time~~ to form a set $J_1, J_2 \cdots, J_n$ such that start time of $(J_a) \le$ start time of $(J_b)$ if $a<b$

① $k=1$

② check all previously used processors
if any processor is empty, assign $J_k$ to it
if no previous processor is empty, assign $J_k$ to a new processor

③ $k$++

④ if $k==n+1$, the algorithm stops
else, go to ②

(b) prove by induction
base case: if there is only one interval that needs to be processed, my algorithm is optimal
induction step: we assume for the intervals $I_1, I_2 \cdots I_m$, my algorithm is optimal
if we have $m+1$ intervals, prove by contradiction.
if we have an optimal algorithm that uses less processors, it has to use more or equal processors before the last interval gets involved. When the last interval gets in, my algorithm either puts it in an empty processor, which is impossible for the optimal algorithm to do better, or assign a new processor to it. If optimal is better, it needs to put it in an empty processor, but if there is no empty one in my algorithm, the only possible way for optimal to do is to previously have more processors, and then some might be empty. contradiction



2. (20 points) (a) Design an efficient algorithm that outputs the vertices of a DAG (Directed Acylic Graph), such that if there is an edge $(x, y)$ then $x$ is output before $y$.

   (b) Analyze the run time of your algorithm.

① for all vertices $V_1, V_2 \cdots V_n$, calculate the number of edges go from it as $f_1 f_2 \cdots f_n$, and edges go in it $i_1, i_2 \cdots i_n$

② find the vertices V that have $i = 0$, output them, call them source

③ for all vertices that there is an edge from source to it, $i$--, delete the ~~sources~~ edges?

④ same as ②, but this time only look for $i = 0$ vertices in the just decreased $i$ vertices

⑤ go to ③

?

9/10

Suppose there are n vertices and e edges

(b)

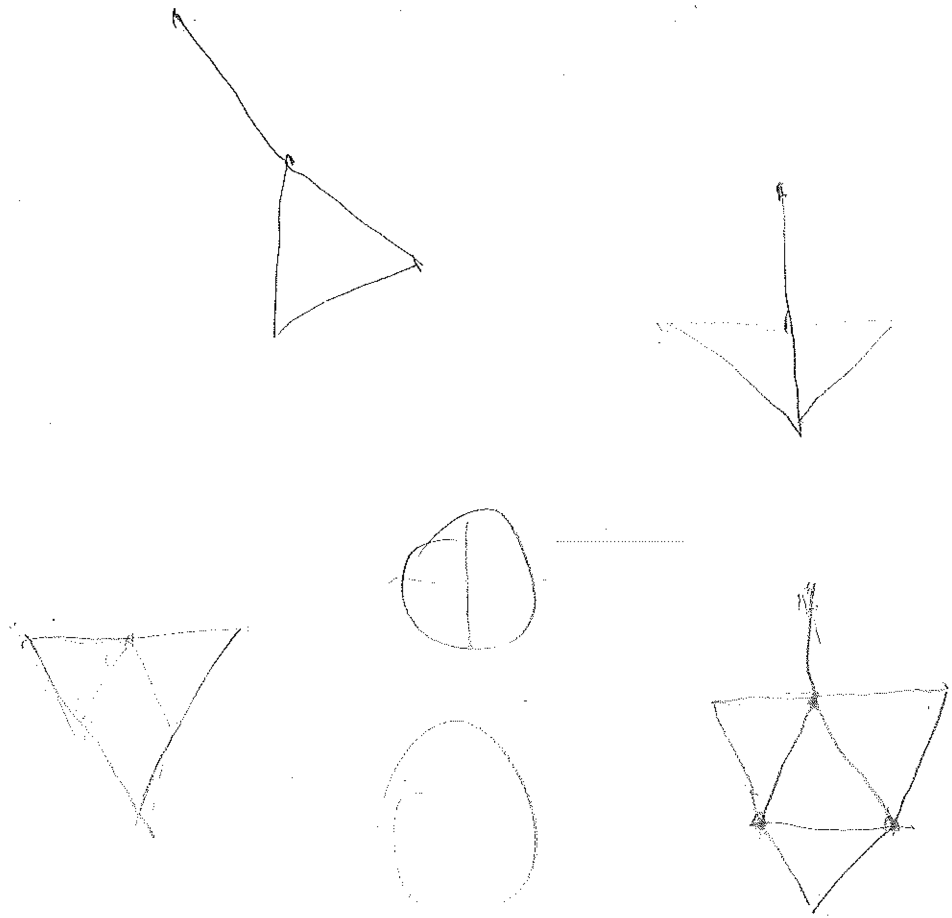
① takes ~~n~~ $O(n+e)$

② takes ~~n~~ $O(n)$

③ ④ ⑤ takes $(n+e)$ in total, because all the edges are searched once and vertices constant time

$\therefore O(n+e)$

2/10



3. (20 points) An undirected graph is said to have property $X$ if you can start from a vertex, traverse all edges of the graph exactly once, without removing your pen from the paper.

   (a) Classify the graphs that have property X?

   (b) Design an efficient algorithm for generating a traversal of a graph that has property $X$.

(a) graphs that have an Euler cycle, that is, graphs that have only 0 or 2 odd degree vertices ✓

(b) ~~① start from an odd degree vertice.~~ if there is, if there is not, ~~start from an~~ arbitrary vertice ~~mark it~~ as current

~~② if there is an unmarked edge connected to current, go to the other vertice it connects to, mark it as current, and mark the edge~~

~~③ go to ②~~

① start from an odd degree if there is, an vertice with maximal degree vertice if there is no odd

-7

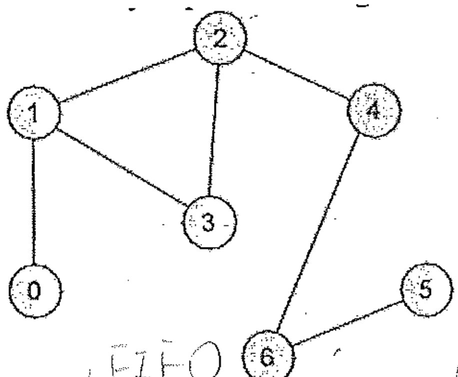
② DFS

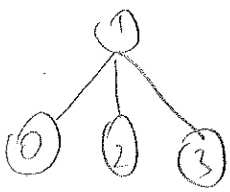
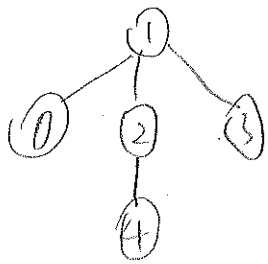
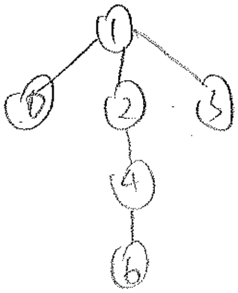
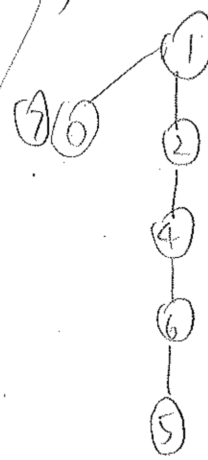
10



4. (10 points) Consider an unweighted graph $G$ shown below:

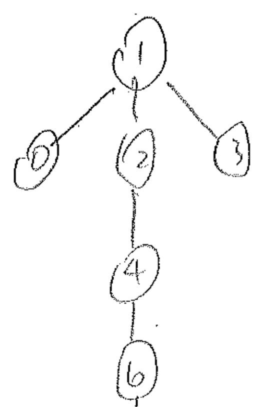
   (a) Starting from vertex 1, show every step of BFS along with the corresponding FIFO next to it.



| step | tree | FIFO ~~stack~~ queue |
|---|---|---|
| 1 | 1 | 1 |
| 2 |  | 0 2 3 |
| 3 | same as 2 | 2 3 |
| 4 |  | 3 4 |
| 5 | Same as 4 | 4 |
| 6 |  | 6 |
| 7 |  | |
| 8 | Same as 7 | (empty) |

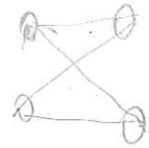
5

the result of BFS is:



5. (20 points) Consider an unsorted list of integers. You can find the minimum number in the list with $n-1$ comparisons. Similarly, you can find the maximum with $n-1$ comparisons. So you can find both the minimum and the maximum with about $2n-3$ comparisons. Design an algorithm that finds both the minimum and the maximum using about $\frac{3n}{2}$ comparisons.

8 5
16 7
32 9
$2^n$ $2n-1$
$(2(\frac{n}{3})-3)$
$+4$



6. (10 points) Give an algorithm to color a graph with 2 colors (assuming it is 2-colorable). A proof of correctness is not necessary.

(1) start from an arbitray vertice

(2) do a breadth-first search

(3) mark vertices with odd depth on the BFS tree one color, and vertices with even depth another color

-4 Time complexity