EECS 660 Fundamentals of Computer Algorithms Final

Name:___________________________ **KUID:**____________________

Grading: 30% towards final grade.
Time: May 11th, 2018, 7:30AM-10:00AM (150mins)

Instructions:

- This is a closed-book, closed-notes exam.
- Please do NOT use pencil; answers written using pencils will NOT be graded.
- Please do NOT write on the back of the paper; answers written on the back will NOT be graded.
- Your writing should be clean, neat, and legible; if we cannot read it, it will NOT be graded.
- You must show all your works clearly for credit; partial credit will only be given to meaningful answers.

1: Recall the minimized maximum lateness problem. You are running a single-thread server where no two tasks can execute concurrently. You are given a list of tasks, each task i comes with a deadline d_i and a requested execution time t_i. You are expected to schedule the tasks such that the maximum lateness among all tasks, i.e. $\max_i(f_i - d_i)$, is minimized (where f_i is the finishing time of the task i in your schedule).

Device an algorithm to find the schedule with minimized maximum lateness. Present in pseudo-code. (5pts)

greedy algorithm
Earliest Deadline first

Order the jobs in order of their deadlines
Assume for simplicity of notation that d1 ≤ . . . ≤ dn
Initially, f = s; //init Job 1
Consider the jobs i=1,...,n in this order //for loop
Assign job i to the time interval from s(i)=f to f(i)=f +ti Let f = f + ti
End
Return the set of scheduled intervals [s(i), f (i)] for i = 1, . . . , n //for loop to print

Text

Prove that all schedules without inversion will have the same maximum lateness. An inversion is defined for two tasks i and j, where $d_i < d_j$ and $s_i > s_j$ (s_i is the starting time of task i in your schedule; in other words, task i is scheduled after j). (10pts)

All tasks with same deadline are sorted and executed consecutively, therefore

问题为啥和这个有关系?

2: Recall the weighted interval scheduling problem. You are running a single-thread server where no two tasks can execute concurrently. You are given a list of tasks, each task i comes with a start time s_i, a finish time f_i, and a profit w_i. Devise an algorithm to determine a schedule that maximize the total profit (include traceback). (10pts)

dynamic programming

3: Devise an algorithm to determine whether a graph $G = (V, E)$ is bipartite. Present your algorithm in pseudo-code. (10pts)

Way1: 着色法
set all nodes are no color.
the start node set red color in the beginning.
while all nodes have color
go to next node
if next node is no color; set opposite color
elseif next node has one color, compare the two node if has different color
else return false

https://blog.csdn.net/li13168690086/article/details/81506044

Way2: check odd or even length cycle
odd - not bipartite
even - bipartite
there are two conditions: even is cycle
odd is not cycle
while each nodes
count = 0
DFS(each node,count)
if count is odd
return false
endif

DFS(i,count){
stack
if(node is visisted, and node == i) return count;
else
count++;
}

4: Given a weighted directed graph $G = \{V, E\}$, and the cost/length for each edge (denote the length of edge e as l_e and $l_e \geq 0$ for all $e \in E$), find the shortest path $d(v)$ from vertex u to v. (5pts)

Prove that your algorithm is correct. (10pts)

5: Present an algorithm that finds the median from a list of unsorted numbers in linear time. (5pts)

p728

Show that your algorithm runs in linear time. (10pts)

6: Prove the master theorem. (hint: the sum of a geometric series is $\sum_{k=0}^{n-1} ar^k = a\frac{1-r^n}{1-r}$) (10pts)

7: Recall the stable matching problem.

Assume that we have n men and n women. Each man gives his preference on the women, and each woman gives her preference on the men. No tie is allowed in the rankings. The stable matching problem seeks to find a set of stable one-to-one pairs between the men and women, such that no instability is present in the set. By instability we mean that there exist two pairs, where both the man in one pair and the women in another prefer each other over their respective partners (and therefore have a strong motivation to break with their partners to form a new pair).

For example, consider pairs (m, w) and (m', w'), where m and m' are two individual men and w and w' are two individual women. If m prefers w' over w, and w' prefers m over m', then m and w' would break with their current partners and pair up, which corresponds to an instability of the matching.

Devise an algorithm to solve the stable matching problem. Your algorithm should be both correct and efficient. (5pts)

Set each person are free
while (some man is free or have not proposed to all women){
m choose the first w preference list
if the w is free — pair(m,w); set not free (m,w)
else if the w has m but w prefer m' over m
set m is free and pair (m', w);not free(m',free)
else w reject m
}

Initially all m∈M and w∈W are free
While there is a man m who is free and hasn't proposed to every woman
Choose such a man m
Let w be the highest-ranked woman in m's preference list
to whom m has not yet proposed If w is free then
(m, w) become engaged
Else w is currently engaged to m′
If w prefers m′ to m then m remains free
Else w prefers m to m′ (m, w) become engaged m′ becomes free
Endif Endif
Endwhile
Return the set S of engaged pairs

Prove that upon the termination of your algorithm, there exists no single man nor woman who is not paired. (10pts)

if upon the termination, there
all man have proposed to all
woman.
if a man who is not unmatched
upon termination, there there is
a women is free.

8: Show that the amortized time complexity for each deletion/insertion (which comes in random order) with table-doubling and table-halving is $O(1)$. (10pts)

Bonus questions:

B1: (The longest common substring problem) Given two strings s and t, find the longest substring that is shared by both s and t. For example, if s=BABA and t=ABAB, their longest common substring is ABA (or BAB). Your algorithm should run in time $O(|s||t|)$. (10pts)

B2: (The minimum spanning tree problem) Given a graph $G = (V, E, W)$ where G is a connected graph, V is the vertex set, E is the edge set, and W is the edge-associated weights ($w(e) \geq 0$ for all $e \in E$). Find a subset of edges $T \subseteq E$ such that T covers (you can traverse between any two vertices only through paths in T) all vertices and the total cost of T $\sum_{e \in T} w(e)$ is minimized. Your algorithm should run in $O(|E| \log |V|)$ time. (10pts)