# 2011 Final

Friday, July 12, 2019 12:19 AM

Final_2011

*Design and Analysis of Algorithms* May 12, 2011
Massachusetts Institute of Technology 6.046J/18.410J
Profs. Dana Moshkovitz and Bruce Tidor Practice Final Exam

# Practice Final Exam

- Do not open this quiz booklet until you are directed to do so. Read all the instructions first.
- The quiz contains 6 problems, several with multiple parts. You have 180 minutes to earn 120 points.
- This quiz booklet contains 15 pages, including this one, and a sheet of scratch paper which can be detached.
- This quiz is closed book. You may use two double sided Letter ($8\frac{1}{2}'' \times 11''$) or A4 crib sheets. No calculators or programmable devices are permitted. Cell phones must be put away.
- Write your solutions in the space provided. If you run out of space, continue your answer on the back of the same sheet and make a notation on the front of the sheet.
- Do not waste time deriving facts that we have studied. It is sufficient to cite known results.
- Do not spend too much time on any one problem. Generally, a problem's point value is an indication of how many minutes to spend on it.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Please be neat.
- Good luck!

| Problem | Title | Points | Parts | Grade | Initials |
|---|---|---|---|---|---|
| 0 | Name | 1 | 15 | | |
| 1 | True or False | 44 | 11 | | |
| 2 | P, NP & Friends | 10 | 1 | | |
| 3 | Taming MAX-CUT | 10 | 3 | | |
| 4 | Spy Games | 10 | 2 | | |
| 5 | Lots of Spanning trees | 25 | 5 | | |
| 6 | Traveling with the salesman | 20 | 3 | | |
| Total | | 120 | | 110 | |

**Name:** George Ong

**Problem 0. Name.** [1 point] Write your name on every page of this exam booklet! Don't forget the cover.

**Problem 1. True or False.** [44 points] (11 parts)

Circle **T** or **F** for each of the following statements to indicate whether the statement is true or false, respectively, and briefly explain why. Your justification is worth more points than your true-or-false designation.

**(a)** T **F** [4 points] If problem $A$ can be reduced to 3SAT via a deterministic polynomial-time reduction, and $A \in$ NP, then $A$ is NP-complete.

$A \in$ NP-complete $\Leftarrow$iff$\Rightarrow$ $A \in$ NP, $A \in$ NP Hard

Hardness($A$) $\leq$ Hardness(3SAT)

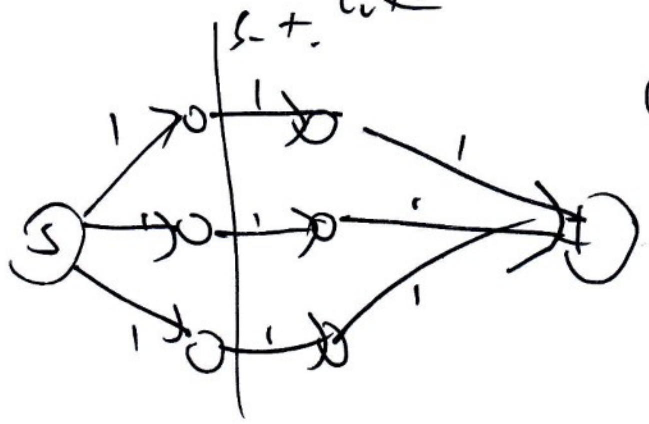
False, we cannot say $A$ is NP-hard (as hard as the hardest NP problems)

**(b)** T **F** [4 points] Let $G = (V, E)$ be a flow network, i.e., a weighted directed graph with a distinguished source vertex $s$, a sink vertex $t$, and non-negative capacity $c(u, v)$ for every edge $(u, v)$ in $E$. Suppose you find an $s$-$t$ cut $C$ which has edges $e_1, e_2, \ldots, e_k$ and a capacity $f$. Suppose the value of the maximum $s$-$t$ flow in $G$ is $f$.

Now let $H$ be the flow network obtained by adding 1 to the capacity of each edge in $C$. Then the value of the maximum $s$-$t$ flow in $H$ is $f + k$.

s-t is a min cut, but may NOT be the only min cut.

Consider

(c) T F [4 points] Let $A$ and $B$ be optimization problems where it is known that $A$ reduces to $B$ in polynomial time. Additionally, it is known that there exists a polynomial-time 2-approximation for $B$. Then there must exist a polynomial-time 2-approximation for $A$.

- $H(A) \leq \max(\text{polynomial}, H(B))$
- $\exists$ $O(n^c)$ - 2-approx for B

A reducing to B provides no such guarantee on PTAS A reducing to PTAS B

(d) T F [4 points] There exists a polynomial-time 2-approximation algorithm for the general Traveling Salesman Problem.

2-approx TSP is also NP-complete

~~TSP p-approx can reduce~~

$\boxed{\text{NP-complete Hamiltonian-cycle can be solved using any hypothetical } \rho\text{-approx TSP}}$

(e) T F [4 points] If we use a max-queue instead of a min-queue in Kruskal's MST algorithm, it will return the spanning tree of maximum total cost (instead of returning the spanning tree of minimum total cost). (Assume the input is a weighted connected undirected graph.)

initialize |V| unary components

Sort edges

For each edge, if nodes belong to diff tree, merge it

(f) T F [4 points] A randomized algorithm for a decision problem with one-sided-error and correctness probability $1/3$ (that is, if the answer is YES, it will always output YES, while if the answer is NO, it will output NO with probability $1/3$) can always be amplified to a correctness probability of 99%.

| return \ shall be | Y | N |
|---|---|---|
| Y | 1 | 2/3 |
| N | 0 | 1/3 |

while(result != N),
~~if R? Y++ : N++~~
if c>100:
return Y
c++
return N

if Y, then Y

Yes. Given independence of our random trial (use of random algorithm) ~~we can repeatedly use the algorithm~~

if N, then ~~(...)~~ $(2/3)^n$ of not NO

**(g)** **T** **F** [4 points] Suppose that a randomized algorithm $A$ has expected running time $\Theta(n^2)$ on any input of size $n$. Then it is possible for some execution of $A$ to take $\Omega(3^n)$ time.

Can

$$E[A] = O(n^2) = p(\Omega(3^n)) + (1-p)\Omega(n^2)$$

hold?

Yes, Suppose $p = \frac{1}{3^n}$

−4

**(h)** **T** **F** [4 points] Building a heap on $n$ elements takes $\Theta(n \lg n)$ time.

build-Heap is $O(n \lg n)$ w/ std. heap algorithm

heapSort is
O(n lgn) because
It requires repeated
heapPops to get
The sorted sequence.

build-Heap is $\Omega(n \lg n)$ because sort can be reduced to heap problem AND sort is $\Omega(n \lg n)$

In-place swapping algorithm that resto
In a sort of divide and conquer / soluti

If I push down a node, any node that p
Of a node I pushed down, because the
Of being bigger than everything below

Work at each level is linearly increasin
Nodes at each level is a geometrically

**(i)** **T** **F** [4 points] We can evaluate a polynomial of degree-bound $n$ at any set of $n$ points in $O(n \lg n)$ time.

We can only do this on a set of n points that ~~~~ are a "n-th degree degree of unity" set.

ores heap invariant
on substructure / greedy way.

percolates UP can never violate the heap invariant
e two candidate nodes have the heap property
y it (we now have a one way ticket to solution!)

g with height
decaying function of height

−4 (j) T F [4 points] Suppose that you have two deterministic online algorithms, $A_1$ and $A_2$, with a competitive ratios $c_1$ and $c_2$ respectively. Consider the randomized algorithm $A^*$ that flips a fair coin once at the beginning; if the coin comes up heads, it runs $A_1$ from then on; if the coin comes up tails, it runs $A_2$ from then on. Then the expected competitive ratio of $A^*$ is at least $min\{c_1, c_2\}$.

competitive ratio

$$E[c] = p c_1 + (1-p) c_2$$

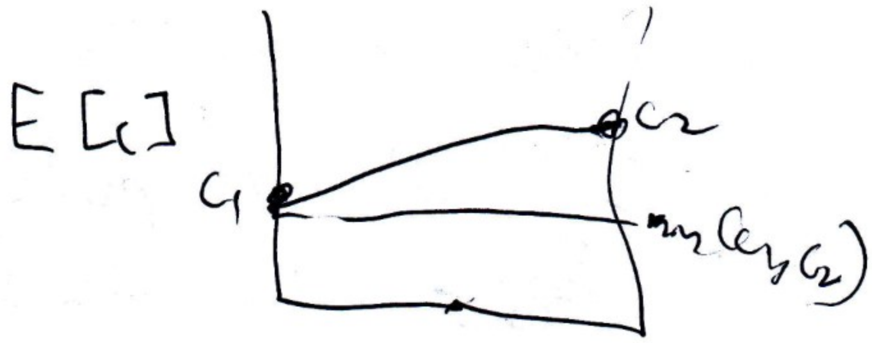
False. When evaluating competititve ratios, it uses an "adversarial input generator" like big O notation, but it has the ability to design inputs that are GOOD for the offline version, and BAD for you.

Our adversary has MORE power when working against a single online algorithm HOWEVER, for a randomized online algorithm, trying to screw algorithm 1 might affect it's Ability to screw algorithm 2. Therefore, expected competitive ratio might be better since

$$\min(c_1, c_2) \le p c_1 + (1-p) c_2 \quad | \; 0 \le p \le 1$$

The adversary cannot scale it's meanness w/ more algorithms in the mix.

$c_1$ ~~+~~ $-c_2$



**Problem 2. Taming Max-Cut** [10 points] A CUT, in a graph $G = (V, E)$, is a partition of $V$ into two non-intersecting sets $A, B$. An edge is said to be in the cut if one of its end points is in $A$ and the other is in $B$. In the MAX-CUT problem, the objective is to maximize the number of edges in the cut. We intend to design an approximation scheme for MAX-CUT. Consider the following scheme. Every vertex $v \in V$ is assigned to $A$, $B$ uniformly at random.

**(a)** What is the probability that $e \in E$ is in the cut?

$e = (u, v) \in E$

$Pr(e \in cut) = Pr(u \in A) \cdot Pr(u \in B) + Pr(u \in B) \cdot Pr(u \in A)$

$= \boxed{0.5}$

**(b)** What is the expected number of edges in the cut?

$0.5|E|$

**(c)** Conclude that the randomized scheme presented above is a 2-approximation to the MAX-CUT.

2-approx

IFF

max weight cut $\leq 2 E[\text{weight of edges in cut}]$

$|E| \leq 2 \cdot 0.5 |E|$

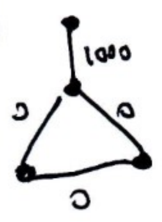
$|E| \leq |E|$ holds

**Problem 3. Lots of Spanning Trees.** (5 parts) [25 points] Let $G = (V, E)$ be a connected undirected graph with edge-weight function $w : E \to \mathbb{R}$. Let $w_{\min}$ and $w_{\max}$ denote the minimum and maximum weights, respectively, of the edges in the graph. Do not assume that the edge weights in $G$ are distinct or nonnegative. The following statements may or may not be correct. In each case, either prove the statement is correct or give a counterexample if it is incorrect.

**(a)** If the graph $G$ has more than $|V| - 1$ edges and there is a unique edge having the largest weight $w_{\max}$, then this edge cannot be part of any minimum spanning tree.

$|E| > |V| - 1 \Rightarrow \exists$ a cycle

~~FALSE~~ ~~TRUE~~ FALSE




$e_{max}$ may be the only edge that connects certain nodes

**(b)** Any edge $e$ with weight $w_{min}$, must be part of some MST.

TRUE

PBC.

Graph is connected $\Rightarrow \exists$ a tree

↳ Suppose a tree does NOT include $e_{min}$

↳ ~~$\exists$ some edge that connects~~
consider the path from $u$ to $v$.

Suppose we remove any edge on this path and replace it w/ $e_{min}$.

This is a tree w/ weight at most MST

**(c)** If $G$ has a cycle and there is unique edge $e$ which has the minimum weight on this cycle, then $e$ must be part of every MST.

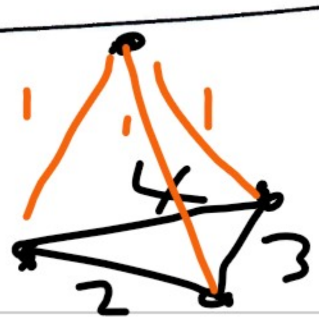
−2

~~TRUE~~

FALSE. We have the option to not use that cycle at all. Suppose it's a cycle with heavy weights, but we have Light weight edges to connect them isntead!

PBC:

Suppose it isn't. If we cut & paste we get a $W(T^*) < W(MST)$



**(d)** If the edge $e$ is not part of any MST of $G$, then it must be the maximum weight edge on some cycle in $G$.

~~FALSE~~


TRUE. otherwise it's a viable cut + paste for an existing MST

**(e)** Suppose the edge weights are nonnegative. Then the shortest path between two vertices must be part of some MST.

~~TRUE~~ FALSE

PBC. counterexample:


**Problem 4. Traveling with the salesman.** [20 points] In the **traveling-salesman problem**, a salesman must visit $n$ cities. Modeling the problem as a complete graph on $n$ vertices, we can say that the salesman wishes to make a **tour** or a hamiltonian cycle, visiting each city exactly only once and finishing at the city he starts from. The salesman incurs a nonnegative integer cost $c(i, j)$ to travel from city $i$ to city $j$, and the salesman wishes to make a tour whose total cost is minimum, where the total cost is the sum of the individual costs along the edges of the tour.

**(a)** Formulate the traveling salesman problem as a language.

is there a tour w/ weight at most k?

TSP = { (G, c, k) : G=(V, E) complete graph

c : maps e ∈ E to a non-neg integer

~~T : ≥0 integer, and~~

~~∃ a tour in G with cost ≤ k~~ }

~~T: tour in G with w(T) ≤ k~~

k : a non-neg integer for which a tour exists w/ weight at most k. }

Language := subset of inputs for which a DP is yes

**(b)** Prove that TSP ∈ NP.

~~Given a tour in G.~~ If the answer is true, you can give me a tour that has that min weight, and I can check it in polynomial time :)

$T = \{e\}$, ~~Ed~~ $\sum_{e \in T} w(e) \le k$?

A **hamiltonian cycle** in a graph is a cycle that visits every vertex exactly once. Define the language HAM-CYCLE $= \{< G >$: there is a hamiltonian cycle in $G\}$.

**(c)** Assuming that HAM-CYCLE is complete for the class NP, prove that TSP is NP-Complete.

HAM-CYCLE can be ~~encoded~~ reduced into a TSP problem by giving a weight of zero to all edges and querying if a 0-weight TSP tour exists.

∴ ~~TSP~~ HAM-CYCLE is at most as hard as TSP, or

TSP is at least as hard as HAM-CYCLE, which is NP hard

∴ TSP $\in$ NP, $\in$ NP-Hard $\Rightarrow$ $\boxed{\text{NPC}}$