NAME and UID: Mario Tinoco !!

404 500 049

CS180 — Algorithms and Complexity
Winter 2015
D.S. Parker, Yuh-Jie Chen, Xiaoran Xu, Garrett Johnston

Midterm Examination
OPEN BOOK, OPEN NOTES
Wednesday, February 11, 4:00–5:50pm

Do not cheat.

Problem	Points
1	21 /25
2	19 /25
3	16 /25
4	23 /25
Total	/100

In any answer on this exam, you can make reference to any definition or result from the [KT] text by giving a section number, page number, gray box number, verbal summary, etc.

There is NO need to provide a complete reproduction or proof of these results. Short answers are good.

Similarly, you can use any definition or result from the course notes, slides, homework assignments, etc. Just be clear in your references to these sources.

A **Master Theorem**: for $a > 1$, $b > 1$, $k \geq 0$, the solution for $T(n) = a\,T(n/b) + c\,n^k$ is

$$T(n) = \Theta(n^\ell) \quad \text{if} \quad k < \ell = \log_b a$$
$$T(n) = \Theta(n^\ell \log n) \quad \text{if} \quad k = \ell = \log_b a$$
$$T(n) = \Theta(n^k) \quad \text{if} \quad k > \ell = \log_b a.$$

A **Minimum Spanning Tree** in an undirected graph with edge costs $G = (V, E, c)$ is a spanning tree T for G that has minimal total cost $\sum_{e \in T} c(e)$.

A **Shortest Path** from a source node s to t in a directed or undirected graph $G = (V, E, \ell)$ with edge lengths ℓ is a path P from s to t with minimal total length $\sum_{e \in P} \ell(e)$.
In this exam, a **Shortest Path Tree** is a directed tree of edges outward from s selected by Dijkstra's algorithm.

useful identities: $\displaystyle\sum_{k=1}^{N} k^p = \frac{1}{p+1} N^{p+1} + O(N^p)$ $\quad \displaystyle\sum_{k=1}^{N-1} a^k = \frac{a^N - 1}{a-1}$ $\quad \displaystyle\sum_{k=1}^{N-1} k\,a^k = \frac{N\,a^N}{a-1} + O(a^N)$

1 The Master Theorem (25 points) −4

Three platypuses meet in a bar and start to argue about the Master Theorem.

✓ (a) Master Theorem? (8 points)

One of the platypuses says that, if we assume that $a > 1$, $b > 2$, $\ell = \log_b a$, and c and k are positive constants, then the recurrence $T(n) = a\,T(n/b) + cn^k$ has solution

$$T(n) = \begin{cases} \Theta(n^\ell) & \text{if } a > b^k \quad \ell > k \\ \Theta(n^k \log n) & \text{if } a = b^k \quad \ell = k \\ \Theta(n^k) & \text{if } a < b^k. \quad \ell < k \end{cases}$$

The other two platypuses laugh and say this is wrong. The first one gets angry and asks you to help prove it. The two laughing platypuses ask you to give a counterexample. What is your answer?

- [x] the first platypus is right, the recurrence is correct.
- [ ] the laughing platypuses are right, the recurrence is incorrect.

Short proof, or counterexample:

which is Master's Thrm Given →

$$\begin{cases} a > b^k \to \ell > k \\ a = b^k \to \ell = k \\ a < b^k \to \ell < k \end{cases}$$

$$T(n) = \begin{cases} \Theta(n^\ell) & \text{if } a > b^k \to \ell > k \\ \Theta(n^\ell \log n) & \text{if } a = b^k \to \ell = k \\ \Theta(n^k) & \text{if } a < b^k \to \ell < k \end{cases}$$

because of logs

$\log_b b^k = k \log_b b = k$

−4 (b) Laughing Theorem (8 points)

One of the laughing platypuses says that the solution of $T(n) = a\,T(n/b) + c \log_b n$ is

$$T(n) = \Theta(n \log n) \qquad \text{(assuming } n \text{ is a power of } b\text{, and } T(1) = O(1)\text{)}$$

and asks you to prove this. The angry platypus says no, and asks for the correct solution. What is your answer?

- [ ] the laughing platypus is right, the solution is correct.
- [x] ✓ the angry platypus is right, the solution is incorrect.

Short proof, or corrected solution:

$\to c \log_b n = cr = c'$ (constant

n is power of $b \to n = b^r \to \log n = r \log b \to r = \log_b n$

$T(n) = a\left(a\,T\left(\frac{n}{b^2}\right) + c'\right) + c'$

$T(T(n)) = a^2\, T\left(\frac{n}{b^2}\right) + 2c' \to a^r\, T\left(\frac{b^r}{b^r}\right) + rc'$

$T(n) = a^{\log_b n}\, T(1) + c' \log_b n = O(\cancel{a^{\log_b n}})$

✓ (c) Time Complexity (9 points)

The platypuses start fighting over the asymptotic complexity $T(n)$ of the following algorithm A:

```
def A(x,y):
    if length(x) == 1:  return f(x,y);
    x1 = first_half(x); x2 = second_half(x);
    y1 = first_half(y); y2 = second_half(y);
    z1 = A( x1, y1 );
    z2 = A( x2, y2 );
    z3 = A( x1+x2, y1+y2 );
    return  z1 + z2 + z3 + f(x,y);
```

recurrence: $T(n)$	=	$3\,T(\frac{n}{2}) + 4\Theta(n)$
$T(1)$	=	$\Theta(n)$
solution: $T(n)$	=	$\Theta(n^{\log_2 3})$

Assume that the inputs x and y are vectors of size n, and the input lengths n are always a power of 2. The functions `first_half` and `second_half` each take an vector of size n as input, and yield an output vector of size $n/2$. The function `f` takes time $\Theta(n)$ to compute if its arguments have length n.

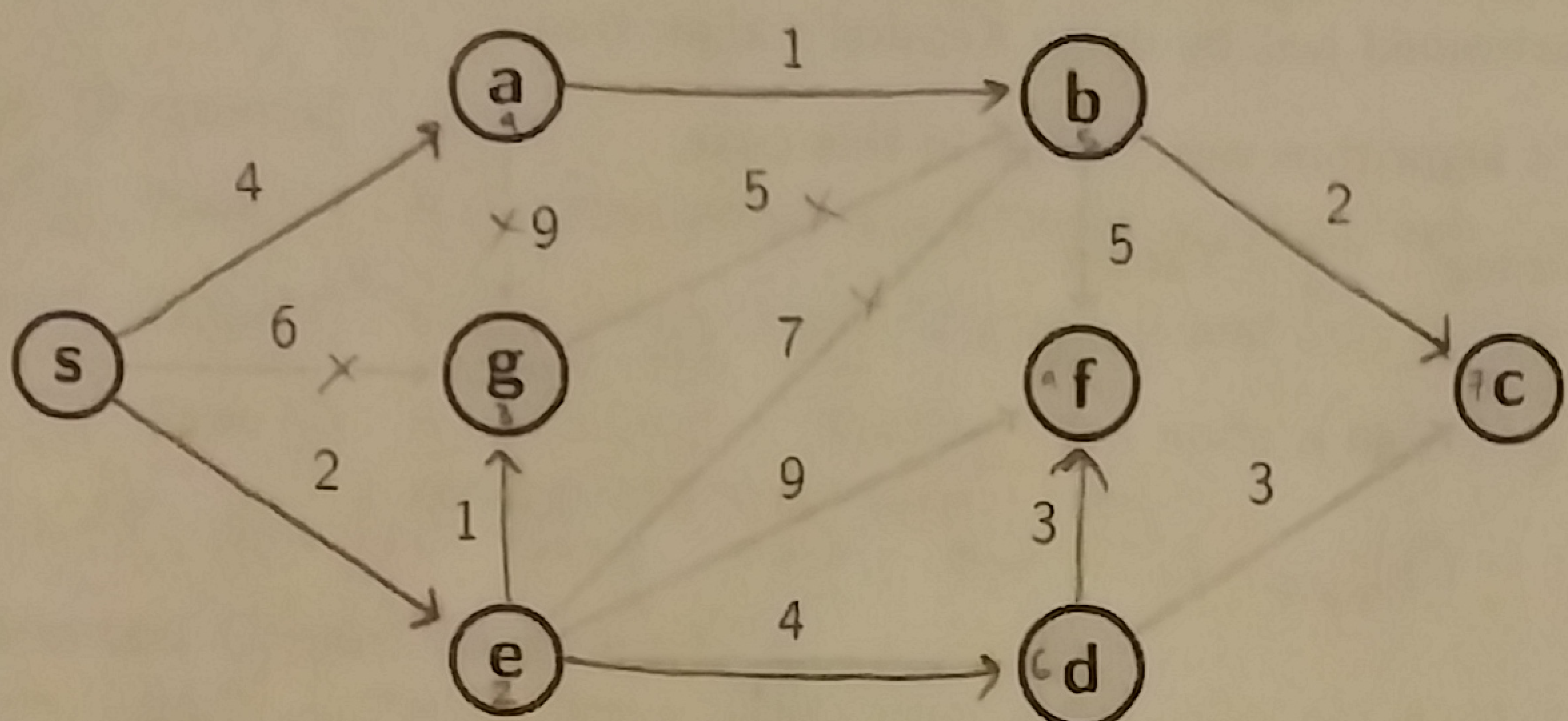
Please stop the fight by providing the recurrence for $T(n)$ and its solution in the box above.

$\ell = \log_2 3 > k = 1$

rtest Paths (25 points) −6

(a) Applying Dijkstra's algorithm (7 points)

Kim Kardashian and Kanye West buy the directed weighted graph G below for $2M. They are trying to apply Dijkstra's algorithm to the graph, starting at vertex s. They want to know what is the order in which vertices get added to the shortest-path tree. Please tell them the order by filling in the box, and draw the resulting shortest-path tree on their graph.



order of addition to shortest-path tree

1	2	3	4	5	6	7	8
s	e	g	a	b	d	c	f

(b) Graph with Distinct Edge Lengths (6 points)

Kim complains that their graph is too small and spends $3M on a larger directed acyclic graph $G = (V, E, \ell)$. In this graph, all edges e have distinct (unique) lengths $\ell(e) > 0$. She asks you whether Dijkstra's algorithm is guaranteed to yield a unique shortest-path tree from any source node s in her new graph.

Your answer is: ☒ Yes ☒ No. OK

Proof:

KT[4.14] At any point in exec of Dijkstras for each $u \in S$ P_u is a shortest s-u path. Since we have unique edge lengths, we will get a unique shortest path

(c) Graph with Negative Edge Lengths (6 points)

Kanye has a life-changing experience and realizes *we all need negative edges*. He buys lots of directed graphs with negative edge lengths, but never buys graphs that have cycles with a negative total length. He asks you: "if I use Dijkstra's algorithm on these graphs, is its resulting shortest-path tree guaranteed to be correct?"

Your answer is: ☐ Yes ☒ No.

Proof:

$P(u) + P' \not\geq P(u)$ not guaranteed

Since edge lengths may be negative, we are not guaranteed that the full path P is at least as long as P_u so the greedy algorithm is not garanteed to be correct

(d) Air Travel (6 points)

−6

HW2 gives you the US airport graph $G = (V, E, \ell)$, and asks you to find a shortest path tree T from LAX. All edge lengths were given as distances in miles, but if we divide by some typical airspeed like 500mph, we can convert the edge lengths into hours. So assume each edge length $\ell(e)$ is given in hours.

Kim complains that your shortest paths are not realistic, since air travel requires at least a one-hour layover in each airport, so you change the length $\ell(e)$ of every edge e in the graph to $\ell'(e) = \ell(e) + 1$ hour.

Is the tree T for G guaranteed to still be a shortest-path tree from LAX for the changed graph $G' = (V, E, \ell')$?

Your answer is: ☒ Yes ☒ No.

Proof:

This is equivalent to adding a constant to all ℓ in which case Dijstras $d'(v) = \min_{e(u,v): u \in S} + \ell_e$ would still result in the same edges being added so we will still get a shortest path tree

3 Minimal Spanning Trees (25 points) −9

Two highly-paid consultants, Kleinberg and Tardos, are arguing about spanning trees in an undirected graph G. You are hired as an even more highly-paid consultant-consultant to resolve their dispute.

(a) MSTs with Negative Costs (9 points)

Assume that all edge costs c are distinct, but the edge costs are permitted to be *negative*.

Kleinberg argues the MST can be determined just by using Kruskal's algorithm.

Tardos says this is ridiculous, Kruskal's algorithm won't work in this case.

Which consultant is right? ☒ Kleinberg ☐ Tardos

Proof:

By KT (4.17) & KT (4.20) Cut and Cycle Property we can guarantee that the cheapest edge will not be in the graph we can then make the appended node the new s and keep recursively adding and get an MST (Kruskal's holds)

(b) Maximum Spanning Trees (9 points)

−9

The company changes its specifications so that all edge costs c must satisfy $c > 1$, and it wants an algorithm to construct **Maximum** Spanning Trees. In other words, it wants an efficient algorithm that finds a spanning tree with the property that the sum of its edge costs is *maximum*.

Kleinberg says that this new Maximum Spanning Tree problem is hard, and will take exponential time to solve.

Tardos says the problem can be easily solved with minor changes to any Minimum Spanning Tree algorithm.

Which consultant is right? ☒ Kleinberg ☒ Tardos

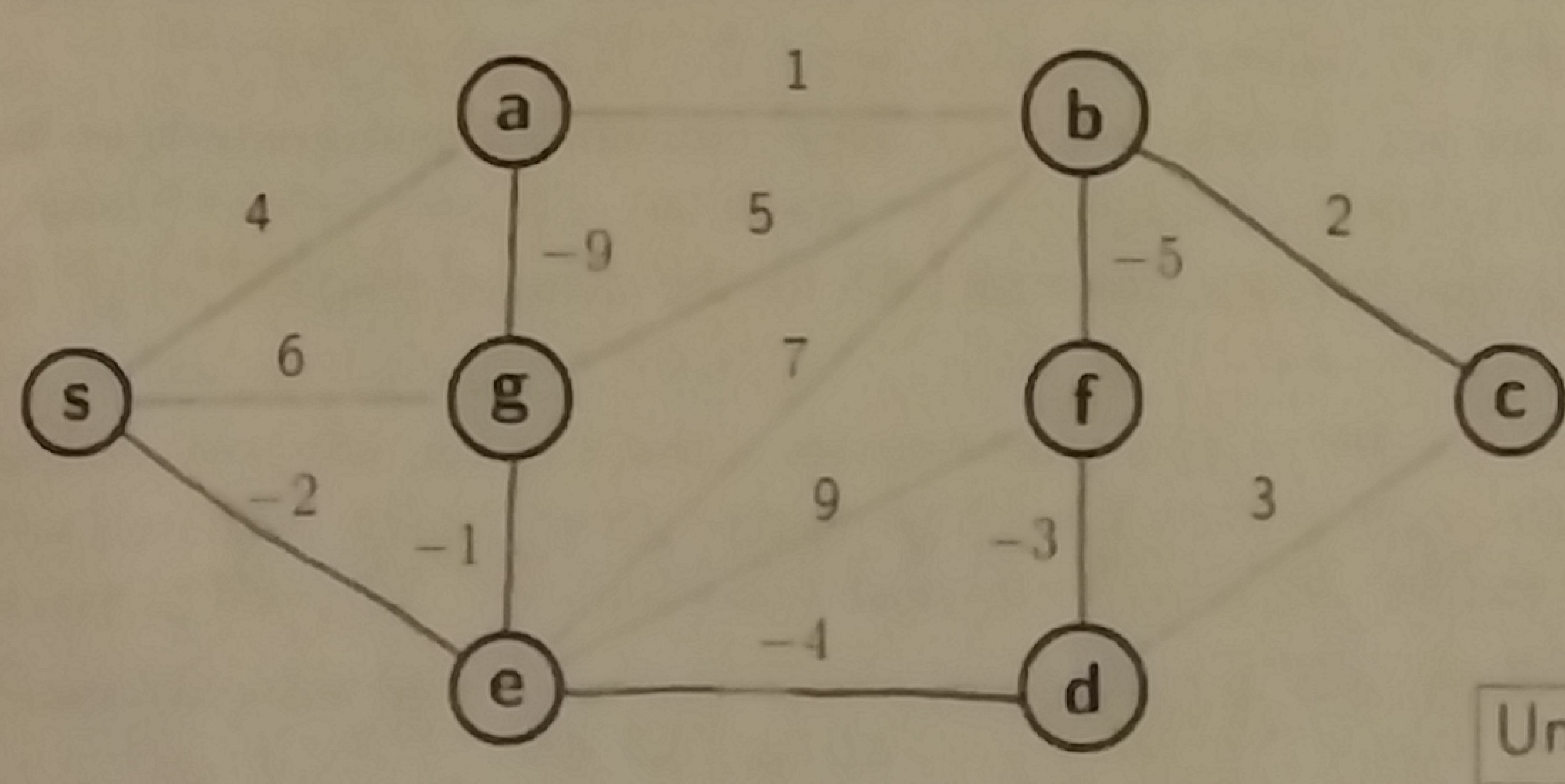
Proof:

The MST Algorithms work because of the cut property and cycle property, however there is no polynomial time equivalent properties for maximum spanning trees.

just use $c(e) \mapsto -c(e)$

(c) Overpaid Consultants (7 points)

Kleinberg and Tardos cannot figure out a MST for the following graph. Please draw a MST for them, and tell them whether or not it is unique.



Unique MST? ☒ Yes ☐ No

rview Questions (25 points) −2

(a) **Changing Minimum Spanning Trees (3 points)**

✓ For an undirected graph G with distinct (unique) edge costs, which of the following statements are true?

True ☐ False ☒ The MST could change if we change the cost of each edge e from $c(e)$ to $c(e) + 2$.

True ☐ False ☒ The MST could change if we change the cost of each edge e from $c(e)$ to $2c(e)$

True ☐ False ☒ The MST could change if we change the cost of each edge e from $c(e)$ to $c(e)^2$

−2 (b) **DAGs (6 points)**

True ☒ False ☐ A directed graph is a directed acyclic graph if and only if it can be topologically sorted

True ☒ False ☒ G is a directed acyclic graph if and only if G has a node with no incoming edges

True ☐ False ☒ A 'DFS tree' T starting from node s in an undirected graph G is sometimes a directed aylic graph that is not a tree.

✓ (c) **Undirected Graphs (8 points)**

True ☐ False ☒ Suppose $G = (V, E)$ is an undirected, connected graph in which all vertices have even degree. Then G is bipartite.

True ☐ False ☒ G is a bipartite graph if and only if G has no triangles (i.e., no three nodes are a clique).

True ☐ False ☒ Suppose a weighted undirected graph G has a cycle C, and there is an edge e that is the unique least-cost edge in C. Then e is in every MST for G.

True ☒ False ☐ If all edge lengths in an undirected graph G are a constant $c > 0$, then for every source vertex s in G, a shortest path tree from s is the same as a BFS tree starting from s.

✓ (d) **Longest Path Problem (4 points)**

In the longest path problem, we're given a weighted directed graph $G = (V, E, \ell)$, and a source $s \in V$, and we're asked to find the longest path from s to every vertex in G.
In general, it's not known whether there is an efficient algorithm to solve the Longest Path problem.
If we restrict G to be acyclic, however, this problem can be solved in polynomial time.
Give an efficient algorithm for finding the Longest Paths from s in a weighted directed acyclic graph G.
(Hint: no incoming edges)

Sort the DAG into a topological ordering then start at set $\{s\}$ and always append the longest edge starting at $\{s\}$ until no V are left to add. At any point in exec. keep an array of edges taken, at any point P_u is longest path to u. where P_u is a backwards traversal of array to s.

✓ (e) **Hamiltonian Path (4 points)**

A Hamiltonian path is a path that visits all nodes in a graph. Explain how, given a *directed acyclic graph* G, it is possible to determine in time $O(V + E)$ whether G has a Hamiltonian path.

turn DAG into topological ordering, $O(V, E)$ (page 103) start at V_1 and travel along topo-order, adding encountered nodes into F if at end F = E then there is a hamiltonian path.