OLD CS 473: Fundamental Algorithms, Spring 2015

Final review...

Lecture 27

May 5, 2015

Multiple choice

Part I

For each of the questions below choose the most appropriate answer.

(A) Given a graph G. Deciding if there is an independent set X in G, such that $G \setminus X$ (i.e., the graph G after we remove the vertices of X from it) is bipartite can be solved in polynomial time.
False: ☐ **True**: ☐ **Answer depends on whether P = NP**: ☐

Multiple choice

Part II

(B) Consider any two problems X and Y both of them in **NPC**. There always exists a polynomial time reduction from X to Y.
False: ☐ **True**: ☐ **Answer depends on whether P = NP**: ☐

Multiple choice

Part III

(C) Given a graph represented using adjacency lists, it can be converted into matrix representation in linear time in the size of the graph (i.e., linear in the number of vertices and edges of the graph).
False: ☐ **True**: ☐ **Answer depends on whether P = NP**: ☐

Multiple choice

Part IV

(D) Given a **2SAT** formula F, there is always an assignment to its variables that satisfies at least $(7/8)m$ of its clauses. **False**: ☐ **True**: ☐ **Answer depends on whether** P = NP: ☐

Multiple choice

Part V

(E) Given a graph G, deciding if contains a clique made out of **165** vertices is NP-Complete. **False**: ☐ **True**: ☐ **Answer depends on whether** P = NP: ☐

Multiple choice

Part VI

(F) Given a network flow G with lower bounds and capacities on the edges (say all numbers are integers that are smaller than n). Assume f and g are two different maximum flows in G that complies with the lower bounds and capacity constraints. Then, the flow $0.7f + 0.3g$ is always a valid maximum flow in G.
False: ☐ **True**: ☐

Multiple choice

Part VII

(G) Given a directed graph G with positive weights on the edges, and a number k, finding if there is simple path in G from s to t (two given vertices of G) with weight $\geq k$, can be done in polynomial time.
False: ☐ **True**: ☐ **Answer depends on whether** P = NP: ☐

Multiple choice

Part VIII

(H) Given a directed graph G with (positive or negative) weights on its edges, computing the shortest walk from s to t in G can be done in polynomial time. **False**: ☐ **True**: ☐ **Answer depends on whether P = NP**: ☐

2: Short Questions.

Part (A)

(A) Give a tight asymptotic bound for each of the following recurrences.

(I) $A(n) = A(n - 3\lceil \log n \rceil) + A(\lceil \log n \rceil) + \log n$, for $n > 2$
and $A(1) = A(2) = 1$.

2: Short Questions.

Part (A) II

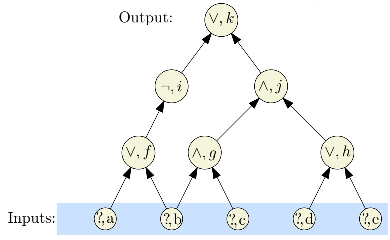
(A) ...

(II) $B(n) = 12B(\lfloor n/4 \rfloor) + B(\lfloor n/2 \rfloor) + n^2$, for $n > 10$ and $B(i) = 1$ for $1 \leq i \leq 10$.

2: Short Questions.

Part (B)

(B) Convert the following boolean circuit (i.e., an instance of **Circuit-SAT**) into a CNF formula (i.e., an instance of SAT) such that the resulting formula is satisfiable if and only if the circuit sat instance is satisfiable. Use $x_a, x_b, x_c, x_d, \ldots$ as the variable names for the corresponding gates in the drawing. (You may need additional variables.) Note, that a node $(\wedge, g)$ in the figure below denotes an and gate, where g is its label.



Scratch slide

3: Balancing vampires.

There are n vampires $p_1, \ldots, p_n$ in Champaign. ith vampire has integer score $w_i \geq 0$ of how well it can climb mountains. Task: Divide the vampires into two teams, and you want the division of teams to be as fair as possible. The score of a team is the sum of the scores of all the vampires in that team. Minimize the differences of the scores of the two teams. Assume that for all i, $w_i \leq W$.

(A) Given integers $\alpha, \beta \geq 0$, and T_α, T_β, such that $\alpha + \beta = n$, describe an algorithm, as fast as possible, to compute the partition into two teams, such that the first team has α players of total score T_α, and the second team has β players with total score T_β. What is the running time of your algorithm? (For any credit, it has to be polynomial in n and W.)
(To simplify things, you can solve the decision version problem first, and describe shortly how to modify it to yield the desired partition.)

Scratch slide

3 (B): Balancing vampires.

(B) Describe an algorithm, as fast as possible, to compute the scores of the two teams in an optimal division that is as balanced as possible, when requiring that the two teams have exactly the same number of players (assume n is even). What is the running time of your algorithm?

3 (C): Balancing vampires.

(C) State **formally** the decision version of the problem in (B), and prove that it is **NP-Complete**. (There are several possible solutions for this part – pick the one you find most natural. Note, that the teams must have the same number of players.)

4: MAX Cut and MAX 2SAT.

Problem: MAX Cut

Instance: Undirected graph G with n vertices and m edges, and an integer k.
Question: Is there an undirected cut in G that cuts at least k edges?

Problem: MAX 2SAT

Instance: A 2CNF formula F, and an integer k.
Question: Is there a truth assignment in F that satisfies at least k clauses.

You are given that **MAX Cut** is **NP-Complete**. Prove that **MAX 2SAT** is **NP-Complete** by a reduction to/from **MAX Cut** (be sure to do the reduction in the right direction!).
Hint: Think about how to encode a cut, by associating a boolean variable with each vertex of the graph. It might be a good idea to verify your answer by considering a graph with two vertices and a single edge between them and checking all possibilities for this case.

Scratch slide

5: Billboards are forever.

Consider a stretch of Interstate-57 that is m miles long. We are given an ordered list of mile markers, $x_1, x_2, \ldots, x_n$ in the range $\mathbf{0}$ to m, at each of which we are allowed to construct billboards (suppose they are given as an array $X[1 \ldots n]$). Suppose we can construct billboards for free, and that we are given an array $R[1 \ldots n]$, where $R[i]$ is the revenue we would receive by constructing a billboard at location $X[i]$. Given that state law requires billboards to be at least $\mathbf{5}$ miles apart, describe an algorithm, as fast as possible, to compute the maximum revenue we can acquire by constructing billboards. What is the running time of your algorithm? Your algorithm has to be as fast as possible.

Scratch slide

6: Best edge ever.

You are given a directed graph G with n vertices and m edges. For every edge $e \in E(G)$, there is an associated weight $w(e) \in \mathbb{R}$. For a path (not necessarily simple) π in G, its **quality** is $W(\pi) = \max_{e \in \pi} w(e)$. We are interested in computing the highest quality walk in G between two given vertices (say s and t). Either **prove** that computing such a walk is **NP-Hard**, or alternatively, provide an algorithm (and **prove** its correctness) for this problem (the algorithm has to be as fast as possible – what is the running time of your algorithm?).

Scratch slide

7: Dominate this.

You are given a set of intervals $\mathcal{I} = \{I_1, \ldots, I_n\}$ on the real line (assume all with distinct endpoints) – they are given in arbitrary order (i.e., you can not assume anything on the ordering). Consider the problem of finding a set of intervals $\mathcal{K} \subseteq \mathcal{I}$, as small as possible, that dominates all the other intervals. Formally, $\mathcal{K}$ **dominates** $\mathcal{I}$, if for every interval $I \in \mathcal{I}$, there is an interval $K \in \mathcal{K}$, such that I intersects K.

Describe an algorithm (as fast as possible) that computes such a minimal dominating set of $\mathcal{I}$. What is the running time of your algorithm? Prove the correctness of your algorithm.

Scratch slide

8: Network flow.

You are given a network flow G (with integer capacities on the edges, and a source s and a sink t), and a maximum flow f on it (you can assume f is integral). You want increase the maximum flow in G by one unit by applying a single augmenting path to f. Naturally, to be able to do that, you must increase the capacity of some of the edges of G. In particular, for every edge $e \in E(G)$, there is an associated cost $\mathrm{cost}(e)$ of increasing its capacity by one unit. Describe an algorithm, that computes (as fast as possible), the cheapest collection of edges of G, such that if we increase the capacity on each of these edges by 1, then one can find an augmenting path to f that increases its flow by one unit. How fast is your algorithm? Provide an argument that explains why your algorithm is correct.