Algorithms Exam [1]

Oct. 26 2013
kl 14 - 18
väg och vatten salar

Ansvarig:
Chien-Chung Huang Tel. 031 772 1699 Rum 6455 EDIT

Points :	60	
Grades:	Chalmers	5:48, 4:36, 3:28
	GU	VG:48, G:28
	PhD students	G:36
Helping material :	course textbook, notes	

- Recommended: First look through all questions and make sure that you understand them properly. In case of doubt, do not hesitate to ask.
- **Answer all questions in the given space on the question paper (the stapled sheets of paper you are looking at). The question paper will be collected from you after the exam. Only the solutions written in the space provided on the question paper will count for your points.**
- Use extra sheets only for your own rough work and then write the final answers on the question paper.
- Try to give the most efficient solution you can for each problem - your solution will be graded on the basis of its correctness *and* efficiency – a faster algorithm will get more credit than a slower one. In particular a brute force solution will not get any credit.
- Answer concisely and to the point. (English if you can and Swedish if you must!)
- Code strictly forbidden! Motivated pseudocode or plain but clear English/Swedish description is fine.

Lycka till!

[1] 2013 LP 1, DIT600 (GU) / TIN092 (CTH).

Problem 1 Vertex Cover [10] Remember the vertex cover problem that we discussed in class: Given $G = (V, E)$, we want to know the minimum size of the vertex cover $C \subseteq V$, where C is a vertex cover if every edge $e = (u, v) \in E$ has at least one of its endpoints u or v in C.
We know the decision question of this problem is NP-complete in general. However, when the graph is "simple enough", the question may not be so difficult: a tree is a rather simple graph.

(a) [5 pts] Suppose that G is a tree. Give a polynomial time algorithm to find the minimum vertex cover. (Hint: Consider being greedy — and remember that you can choose the optimal solution you like.)

(b) [5 pts] Prove that your algorithm is correct.

Problem 2 Organizing the Town Council [10] Suppose that Thule has r residents, $R_1, \cdots, R_r$; c clubs $C_1, \cdots, C_c$, and p political parties $P_1, \cdots, P_p$. A resident R_i can belong to multiple clubs but only one particular party. We want to know the answer to the following question: is it possible to let each club C_j choose one of its members to represent it in the town council so that (1) no resident is a representative of two (or more) clubs, and (2) each party P_k has at most u_k representatives in that town council (remember that a chosen resident also belongs to a specific party).

(a) [4 pts] We want to know whether it is possible to organize such a town council. Show that we can solve the problem in polynomial time by formulating this prob-

lem as a max-flow network problem. You should describe the network explicitly, i.e., the source, the sink, and all the edge capacities.

(b) [6 pts] Argue that we can organize such a town council if and only if your network has a certain flow value (give the specific value explicitly). Notice that if you claim that the town council is possible, you should explain how to organize it using your maximum flow.

Problem 3 Relaxed Inversion [10] Recall that we have discussed how to use the divide-and-conquer strategy to solve the inversion problem. Remember in this problem, we are given an array $A[1 \cdots n]$, and we need to count the number of inversions: a pair $i < j$ is an inversion of $a_i > a_j$. (For the sake of simplicity, let us assume that all numbers in the array A are distinct). Let us consider a slight variation of this problem.

We count a pair $i < j$ as an inversion only if $a_i > 4a_j$. Show that this problem also can be solved in $O(n \log n)$ time as originally.

(a) [2 pts] Describe how we do the "Divide–and–Conquer" step. What kind of outcome do you want from the subproblem?

(b) [5 pts] Describe how we do the "Combination" step. In particular, show that this step can be done in linear time.

(c) [3 pts] Write a recurrence for the running time of the algorithm. Show the actual running time of the algorithm. You can assume that n is some power of 2.

Problem 4 Minimum Bounded-degree Spanning tree [10] In class we have discussed the minimum spanning tree problem. This problem can be solved in polynomial time. Now let us consider a generalization of the minimum spanning tree problem. Given $G = (V, E)$ and a cost function $c : E \to R_{\geq 0}$, and a *degree constraint*, $b : V \to Z_{\geq 0}$, is there a spanning tree T of total cost at most C, under the following condition: each vertex v in the tree T cannot have more than $b(v)$ neighbors, i.e., $|\{u|(u, v) \in T\}| \leq b(v)$.

(a) [3 pts] Show that this problem belongs to the class of NP.

(b) [7 pts] Show that this problem is NP-complete. (Hint: (undirected) Hamiltonian path sounds a good candidate for your reduction.) You should argue that your reduction is valid.

Problem 5 Eulerian Tour [10] Given a *connected* graph $G = (V, E)$ where each vertex $u \in V$ has even degree, i.e., it has an even number of edges in E incident on it. It is known that such a graph allows an *Eulerian tour*: there is a edge-disjoint tour P so that P goes through all edges exactly once (but P can visit the same vertex more than once) and P has the same beginning and ending vertex.

There are a few ways of proving this. Below we try to prove by induction on the number of the edges $|E|$. The base case is $|E| = 0$ so it is vacuously true. The induction hypothesis states that when there are less than k edges in the graph, there is such an Eulerian tour. Let us think about the induction step.

(a) [1 pts] First show that if the graph G is connected and all vertices have even degree, then there is always a cycle C of length at least 2 contained in this graph.

(b) [5 pts] Use the fact that the graph G has such a cycle C to prove that the graph G must have an Eulerian tour. (Hint: if we remove the cycle C from the current graph G, then the number of edges are decreased—then could we not apply the induction hypothesis? But be careful. After C is removed from G, the graph can be splitted into several connected subgraphs $G_1, \cdots, G_k$. You should be careful when you reconstruct the entire Eulerian path in G.)

(c) [4 pts] Turn that above proof into an algorithm to find such a Eulerian tour. You should also argue that your algorithm takes only polynomial time.

Problem 6 Weighted Independent Set [10] Remember the independent set problem that we discussed in class: Given $G = (V, E)$, we want to know the maximum size of the independent set $S \subseteq V$, where S is an independent set if no two vertices in S are connected by an edge in E.
We know the decision question of this problem is NP-complete in general. However, when the graph is "simple enough", the question may not be so difficult: a line is a rather simple graph.

(a) [1 pt] Show that there is a polynomial time algorithm to find a maximum independent set if the given graph is just a line.

(b) [2 pts] Now let us consider a more general version of the independent set problem: assume that there is a weight function $w : V \to R_{\geq 0}$. Suppose that the given graph is just a line. The goal is to find a subset of vertices $U \subseteq V$ so that $\sum_{u \in U} w(u)$ is maximized under the condition that U is an independent set.

It turns out that this problem can be solved by dynamic programming. Suppose that vertices along the line is numbered from 1 to n. Define OPT(i) as the maximum weight independent set drawn from $\{1, 2, \cdots i\}$.

What is OPT(1), the base case?

(c) [5 pts] And how do you write the recurrence for OPT(i), for $i > 1$? You should explain the recurrence as well.

(c) [1 pts] After you find out all the values of OPT(i), what is the answer to the original problem?

(d) [1 pts] What is the running time of your algorithm?