Algorithms Exam [1]

Jan. 2 2015
kl 8:30 - 12:30
väg och vatten salar

Ansvarig:
Bapi Chatterjee Tel. 031 772 1024 Rum 5103 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.
- Exam review dates: TBA

Lycka till!

[1] 2014 LP 1, DIT600 (GU) / TIN093 (CTH).

Problem 1 Stable Marriage [7] Recall the original stable marriage problem that we discussed in the lectures, where each person has a strictly-ordered complete preference over all members of the other gender.
Let us consider the following variation of the stable marriage problem. Each person has a preference list as before. But the preference list needs not be complete (she/he can declare someone to be totally unacceptable). A boy b puts girl g on his list if and only if girl g puts b on her list. Furthermore, ties are allowed on the preference lists (she/he can say multiple persons are equally good).
In this context, we define a *blocking pair* as follows. A boy b and a girl g are a blocking pair in a matching M if any of the following conditions holds.

- Boy b and girl g are both matched in M. b *strictly* prefers g to his assigned girlfriend $M(b)$. g *strictly* prefers b to her assigned boyfriend $M(g)$.
- Boy b is unmatched but girl g is matched in M. g is on the preference list of b and g *strictly* prefers b to her assigned boyfriend $M(g)$.
- Boy b is matched but girl g is unmatched in M. b *strictly* prefers g to his assigned girlfriend $M(b)$ and b is on the preference list of g.

A matching M without a blocking pair is stable.

Now give a polynomial time algorithm to find a stable matching. Argue that the found matching is really stable. Show the running time of your algorithm. (Hint: Some variation of the Gale-Shapley algorithm should work.)
(Sketch of the) Solution
Break the ties arbitrarily on both boys and girls' preference lists and apply the G-S algorithm. The running time is $O(m)$, the total size of all preferences. Let M be the outcome matching. We argue that there is no blocking pair of the first type. Suppose not and (b, g) is one such pair. As b strictly prefers g, he must have been rejected by g at some point (how the tie-breaking is done is irrelevant). g has done that because at that point she had a boyfriend b' who ranks at least as high as b (again the tie-breaking is irrelevant), and her final boyfriend $M(g)$ ranks at least as high as b'. So (b, g) cannot be a blocking pair. The blocking pairs of the other two types can be argued similarly.

Problem 2 Smith Rule [8] Suppose that we are to schedule a set J of jobs on a single machine. Each job j has a processing time p_j and a weight w_j. Jobs are processed on the machine non-preemptively (that is, a job, once started, must be finished before another new job starts). Our goal is to minimize the *sum of the weighted completion times*. That is, we want to minimize $\sum_{j \in J} w_j C_j$, where C_j is the completion time of job j. For example, assume that $p_1 = 3$, $p_2 = 2$, $p_3 = 6$ and if we process by the order of job 1, job 2, and then job 3, then $C_1 = 3$, $C_2 = 5$, and $C_3 = 11$.

It is known that this problem can be solved by a greedy algorithm, often called the *Smith rule*, which works by ordering the jobs based on the increasing ratio p_j/w_j. Prove that this strategy gives the optimal solution. For simplicity, let us assume that all ratios p_j/w_j are distinct.

(Sketch of the) Solution

It suffices to argue that in the optimal solution, if job α immediately precedes job β, then $\frac{p_\alpha}{w_\alpha} < \frac{p_\beta}{w_\beta}$. Suppose not. Then $\frac{p_\alpha}{w_\alpha} > \frac{p_\beta}{w_\beta}$. The cost of the two jobs together is

$$w_\alpha(p_\alpha + X) + w_\beta(p_\beta + p_\alpha + X), \tag{1}$$

where X is the starting of job α. Let us reverse the order of the two jobs (note that then the cost of all other jobs remain unchanged) and the new cost of the two jobs is then

$$w_\alpha(p_\alpha + p_\beta + X) + w_\beta(p_\beta + X). \tag{2}$$

Observe that the expression in (2) is smaller than (1) since $\frac{p_\alpha}{w_\alpha} > \frac{p_\beta}{w_\beta}$, a contradiction.

Problem 3 Recurrence [5] Solve the following recurrence: $T(n) = 6T(n/4) + n$ and $T(0) = 0$. Give the exact expression of $T(n)$. Prove the correctness of your expression. You can assume that n is a power of 4.

(Sketch of the) Solution

This can be shown by drawing the recursion tree. The root has the work of n, the second level has $6(n/4)$ the third level has $6^2(n/4^2)$, and so on. Together we have

$$T(n) = n \sum_{i=0}^{\log_4 n} (\frac{6}{4})^i = O(n).$$

Problem 4 Flow and Cut I [16] Let $G = (V, E)$ be a directed simple graph with capacity $c : E \to R_{>0}$. Two special vertices s and t are the *source* and the *destination*.

(a) [8 pts] Suppose that all edge capacities $c(e)$ is a multiple of α. Is it true that the *value* of the maximum flow is also a multiple of α? And is it true that in a maximum flow f, the flow $f(e)$ on edge e is always a multiple of α? For both questions, if your answer is yes, justify it; if your answer is no, show a counter-example. (Hint: Think carefully how we prove the max-flow-min-cut theorem in the lectures.)

(Sketch of the) Solution

The maximum flow has value a multiple of α. Consider the moment that the Ford-Fulkerson stops when there is no augmenting path in the residual graph. Let G' be the subgraph of G reachable from s in the residual graph. The flow value is exactly the sum of the capacity of the "forward arcs" from G' to the remaining graph. Since each forward arc has capacity the multiple of α, so would the flow value be.

However, in a maximum flow, some edges may have flow that is not a multiple of α. Consider a network with three vertices s, u, and t. There are two parallel arcs from s to u, a single arc from u to t, all of which have capacity 1. There is a maximum flow with 0.5 on both parallel arcs from s to u and 1 from u to t.

(b) [8 pts] Suppose that we have a set of students S_1,$\cdots$, S_s, a set of projects P_1 , $\cdots$, P_p, a set of teachers T_1, $\cdots$, T_t. A student is interested in a subset of projects. Each project p_i has an upper bound $K(p_i)$ on the number of the students that can work on it. A teacher T_i is willing to be the leader of a subset of the projects. Furthermore, he/she has a upper $H(T_i)$ on the number of the students he/she is willing to supervise in total. We assume here no two teachers are willing to supervise the same project. The decision problem here is whether there is really a feasible assignment without violating any of the constrains in K and in H.

Solve the decision problem. Prove the correctness of your algorithm. (Hint: Max-Flow network should help.)

(Sketch of the) Solution

Create a vertex for each student and each teacher. For each project i, create two vertices P_i^{in} and P_i^{out}. A source vertex has an arc of capacity 1 to each student vertex. A student has an arc of capacity 1 to P_i^{in} if he is interested in project P_i. Draw an arc from P_i^{in} to P_i^{out} with capacity $K(P_i)$. An arc of infinity capacity is drawn from P_i^{out} to T_j is the latter is willing to supervise that project. Finally, there is an arc of capacity $H(T_j)$ from teacher T_j to the sink.

There is a feasible assignment if and only if the constructed network has the maximum flow value of s, the number of students. The correctness follows from the observation that a flow of value s is a feasible assignment and vice versa.

Problem 5 Subset Sum [12] The following is the SUBSET SUM problem. Given a set of numbers $A = \{a_1, a_2, \cdots, a_n\}$, is there a subset of numbers $A' \subseteq A$ so that their sum $\sum_{a_t \in A'} a_t$ is exactly C.
This problem can be solved by dynamic programming. Let us build a table P, where the entry $P(i, j)$ means: "is there a subset of $\{a_1, a_2, \cdots a_i\}$ so that their sum is exactly j"?

(a) [4 pts] How do we fill in the entry $P(i, j)$? Write down the recurrence.

(Sketch of the) Solution
For the base case, let $P(0, j_{>0})$ be false and $P(i_{>0}, 0)$ be true.

For recursion, $P(i, j)$ is true if $P(i-1, j)$ is true and, when $C - a_j > 0$, if $P(i-1, C-a_j)$ is true, then let $P(i, j)$ be true, Otherwise, it is set to false.

(b) [2 pts] The number of the rows in the table P is clear—just n. But how many columns are necessary? Justify it.

(Sketch of the) Solution

The number of columns needed is C, since this is the required solution.

(c) [2 pts] After you fill in all the entries, how do you find the answer to the original problem? (Notice that you only need to answer yes or no. If the answer is yes, you do not need to tell what that exact subset is.)

(Sketch of the) Solution
Check the entry $P(n, C)$.

(d) [4 pts] What is the running time of your algorithm? Is it really polynomial? If not, under what condition will your algorithm be really polynomial?

(Sketch of the) Solution
It takes $O(nC)$ time. This is pseudo-polynomial. But it is polynomial if $C = O(n^{O(1)})$.

Problem 6 NP-Completeness [12] For the following two problems, you should argue that (1) they belong to the class of NP, and then (2) prove that they are NP-complete by reduction.
Let $G = (V, E)$ be a graph. A valid *coloring* means that we assign a color to each vertex and no two neighboring vertices have the same color. Of course there is always a valid coloring—unless we put a limit on the number of colors we are allowed to use. In the k-coloring problem, we are asked to decide whether there is a valid coloring using at most k colors. It is known that 3-coloring problem is NP-complete.

(a) [6 pts] Show that the 5-coloring problem is NP-complete. (Hint: Apparently, the reduction could be from the 3-coloring.) **(Sketch of the) Solution**
Let G be the given instance. Create H by adding two extra vertices u and v to G. The two vertices are connected by an edge and they are also connected to all vertices originally in G. G has a 3-coloring if and only if H has a 5-coloring.

(b) [6 pts] Cycle packing means that we use a set of vertex-disjoint cycles $C_1, C_2, \cdots$ to cover all vertices in V (Every vertex is covered exactly once). Suppose that we impose the additional condition that each cycle must be of length at least L. The decision problem here is whether there is a cycle packing satisfying the additional constrain that each used cycle is of length at least L. Prove that this problem is NP-complete. (Hint: Hamiltonian Cycle seems a good candidate).

(Sketch of the) Solution
Cycle packing is a special case of Hamiltonian cycle. We can set L to be $|V|$, the number of vertices in the graph. The graph has a cycle packing satisfying the length constrain if and only if it has a Hamiltonian cycle.