Data Structures and Algorithms
FR 6.2 Informatik

Sanders, Telikepalli WS 03/04

http://www.mpi-sb.mpg.de/~sanders/courses/algdat03/index.html

---

Solutions to assignment 11

Exercise 1
In this exercise we discuss how to convert a minimization problem into a decision problem. Assume that the cost function has nonnegative integer values and that we know an upper bound $\hat{f} \geq f^*$ for the value of an optimal solution. Further assume that we have a program $\mathcal{P} : \mathbb{N} \to \mathcal{L} \cup \{\perp\}$ so that $\mathcal{P}(b)$ outputs a feasible solution $\mathbf{x} \in \mathcal{L}$ with $f(\mathbf{x}) \leq b$ if such a solution exists and signals failure $\mathcal{P}(b) = \perp$ otherwise.

(a) Explain how to find an optimal solution using $O(\log \hat{f})$ calls of $\mathcal{P}$.

Solution
We need to find the smallest $b \in \mathbb{N}$, such that $\mathcal{P}(b) \neq \perp$. We know that $0 < b \leq \hat{f}$. Hence we can do a binary search: We check $P(\hat{f})$, then $P(\frac{\hat{f}}{2})$ etc. Whenever $\mathcal{P}(b) = \perp$, we choose the upper half, else we choose the lower half. If we found b, such that $(\mathcal{P}(b) \neq \perp) \ \wedge \ (b = 0 \ \vee \ P(b-1) = \perp)$, then $b = f^*$ and $\mathcal{P}(b)$ is an optimal solution. The number of calls is in $O(log(\hat{f}))$.

(b) Refine the procedure from part (a) so that it works using $O(\log f^*)$ executions of $\mathcal{P}$, even if we do not know an upper bound of f^*.

Solution
We need to find an upper bound to f^*. We can check $\mathcal{P}$ for the powers of two. The first power of two, for which $\mathcal{P}$ does not return $\perp$, is an upper bound. Then we can proceed as above. The number of powers of two smaller than f^* is in $O(log(f^*))$. Even together with the calls of the first part, the total number of calls is still in $O(log(f^*))$.

Exercise 2
Suppose we wish to invest 14,000 euros for one year. We have identified four investment opportunities. Investment 1 requires an investment of 5,000 euros and gives a profit of 800 euros; investment 2 requires 7,000 euros and gives a profit of 1,100 euros; investment 3 requires 4,000 euros and gives a profit of 600 euros; and investment 4 requires 3,000 euros and gives a profit of 400 euros. Into which investments should we place our money so as to maximize our total profit? Model the problem as a knapsack problem. (Hint: For convenience, drop extra zeros.)

Solution
$\mathbf{p} = (8, 11, 6, 4)$, $\mathbf{w} = (5, 7, 4, 3)$, $M = 14$

Exercise 3
Consider the following algorithm for the traveling salesman problem for undirected graphs with triangle inequality (see the lecture slides for a definition):

- Let T denote the edges of a minimum spanning tree of G.
- Now consider the directed graph $G' = (V, T')$ where for each edge $\{u, v\} \in T$, T' contains the two directed edges (u, v) and (v, u). Note that each node in (V, T') has the same in-degree as out-degree.

- Find an Euler tour C' in G'.
- Convert C' into a simple cycle visiting all nodes by introducing *shortcuts*, i.e., repeatedly apply the following shortcutting step: If there are subsequent edges (u, v) and (v, w) in C such that v is also visited before, replace these two edges by the edge (u, w).

Now we ask you to show that this algorithm yields a two-approximation of the optimal TSP tour. (Hint: you might need to show that the weight of T is no more than the weight of an optimal traveling salesman tour.) Do not forget to argue the above algorithm works in polynomial time.

Solution
Let $C^\star$ be an optimal traveling salesman tour and let T denote the edges of a minimum spanning tree of G.

Then, we have $w(T) \leq w(C^\star)$. *Proof*: Remove one arbitrary edge of $C^\star$ and obtain a spanning tree $T^\star$ with $w(T^\star) \leq w(C^\star)$ (because we have removed one edge with non-negative weight). As T is a *minimum* spanning tree, we have $w(T) \leq w(T^\star)$. These inequalities lead to $w(T) \leq w(C^\star)$. $\square$

Let C' be an Euler tour in G'. As every edge of T is used in both directions, we have $w(C') = 2 \cdot w(T)$. During the last step of the algorithm, two edges (u, v) and (v, w) are replaced by (u, w) if v has been visited before. We get a simple cycle C. Due to the triangle inequality, shortcutting can only decrease the tour length so that $w(C) \leq w(C')$. When we combine all (in)equations, we obtain $w(C) \leq w(C') = 2 \cdot w(T) \leq 2 \cdot w(C^\star)$.

The computation of a minimum spanning tree (step 1) and an Euler tour (step 3) can be done in polynomial time with the algorithms presented in the lecture. For most graph representations, step 2 is for free as an undirected graph is already represented as a *bidirected* graph. Step 4 can be done in linear time: just follow the Euler tour and store in an array which vertices have already been visited; thus, we can decide in constant time for each vertex of the tour if we should eliminate it by shortcutting. As all four steps work in polynomial time, the whole algorithm works in polynomial time.