Midterm 2 Solutions

Problem 1 (30 points)

You are given a sequence of n positive integers, $a_1, \ldots, a_n$, say $(4, 5, 3, 2, 1)$. You are asked to choose among them a set of numbers *no two of which are adjacent*, so that the sum is as large as possible. In the present case, the correct answer is $4, 3, 1$. You want to solve this problem by dynamic programming, with prefixes of the sequence as subproblems.

- Complete the following subproblem definition: $S(i)$ is the largest sum…

  I'll do both of two valid solutions: Either (1) "using only indices 1 through i", or (2) "using indices 1 through i and ending at i".

(a) $S(i+1) = \max\ldots$ either (1) $\max(S(i), S(i-1) + a_{i+1})$ or (2) $\max(S(i-1), S(i-2)) + a_{i+1}$

(b) What is the running time of the algorithm? No explanation needed. $O(n)$

(c) Explain very briefly how you would actually recover the optimum solution.

Let $t[i]$ be the j such that $S(i)$ used $S(j)$ (either $i-1$ or $i-2$ for (1) or $i-2$ or $i-3$ for (2)). For (1), starting at $i = n$ add a_i if $t[i] == i-2$, then set $i = t[i]$ and continue. For (2), first find the max $S(i)$ and then starting at that i, add a_i and set $i = t[i]$ and continue.

(d) Suppose now that each number a_i comes with its *reach* r_i such that, if a_i is included in the sum, then any other number between $i - r_i$ and $i + r_i$ must be excluded. That is, the original problem was the $r_i = 1$ case. Write the formula for $S(i+1)$ now.

Let $p(i)$ be the largest $j < i$ for which $j + \max(r_i, r_j) < i$. The we have:
(1) $S(i+1) = \max(S(i), S(p(i+1)) + a_{i+1})$ or (2) $a_{i+1} + \max(S(p(i+1)), S(p(i+1)-1))$.

(e) What is the running time of the algorithm now? No explanation needed. $O(n^2)$

(f) In the original problem ($r_i = 1$) show by a counterexample that the greedy algorithm (there *is* an obvious greedy algorithm) would fail to solve the problem.

Two obvious greedy algorithms:
1. Take odds or evens: $9, 1, 1, 9$, greedy says 10, OPT $= 18$.
2. Take the highest, exclude its neighbors, repeat: $8, 9, 8$, greedy says 9, OPT $= 16$.

(g) **(Extra Credit!)** Show that the greedy algorithm always returns a solution that is at least half the optimum.

1. Odds or evens: Certainly OPT $\leq \sum_i a_i$. Let S_o and S_e be the sum of the odds and evens, respectively. Then $S_o + S_e = \sum_i a_i \geq$ OPT, so clearly one of them must be at least OPT/2.

2. Exclude neighbors: Our goal is to show at at any step (e.g. every time greedy chooses a number), the greedy solution is at least half of the optimum solution when restricted at the numbers picked and/or excluded so far by greedy. Consider the number picked by the greedy algorithm: it is at least half the sum of its neighbor(s). If it also belongs to the optimum solution, we are fine. If it doesn't, the optimum will either have one or both of the neighbor(s), but in either case the sum will be at most twice that of the greedy element. Repeat this reasoning on the elements left after removing the number chosen by the greedy algorithm and its neighbor(s).

Problem 2 (10 points)

Consider these points on the (x_1, x_2) plane: $(0,0), (0,1), (a,1), (a+1,0)$.

(a) Suppose for now that $a > 0$. Write the linear program that has as set of feasible solutions the quadrilateral defined by these points, and of which both the last two points are optima.

$$\max x_1 + x_2$$

$$x_1 + x_2 \leq a + 1$$

$$x_2 \leq 1$$

$$x_1, x_2 \geq 0$$

Many people included the constraint $x_1 \leq a+1$. This is not necessary, but since we did not ask for the *simplest* linear program we gave full credit to this solution as well (the penalty was that you'd then have to deal with a more complicated dual...)

(b) Write the dual program. What is the optimum point?

$$\min (a+1)y_1 + y_2$$

$$y_1 \geq 1$$

$$y_1 + y_2 \geq 1$$

$$y_1, y_2 \geq 0$$

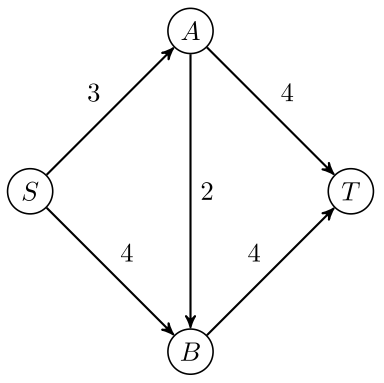
The constraint $y_1 \geq 0$ can be skipped ofcourse. People who got the wrong primal, but a correct dual of the wrong primal, got full credit. The optimal **point** is $(y_1, y_2) = (1, 0)$.

(c) What happens to the primal in question (a) if $a < -1$? To the dual in question (b)?

Primal becomes infeasible, dual becomes unbounded.

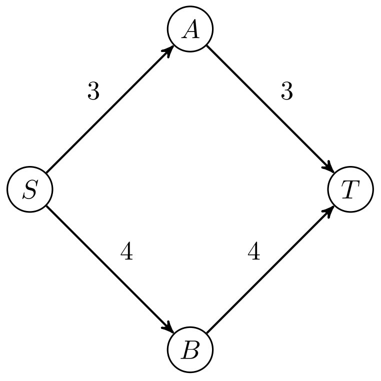
Problem 3 (20 points)

(a) Find the maximum flow and the minimum cut in this network.

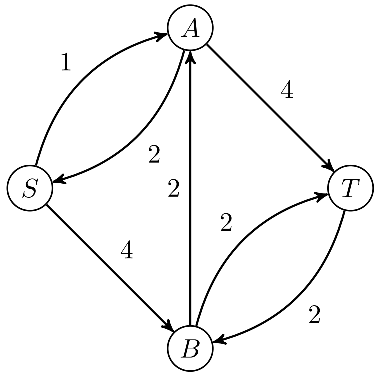


max flow(draw it):

edges of the min cut: (SA,SB)



(b) Suppose that in the first augmentation you choose the longest augmenting path (three edges). What is the residual network after this? (draw it)



(c) Call an edge *crucial* if decreasing its capacity by one would decrease the max flow by one. What are the crucial edges in this network? **Answer:** SA,SB and BT

(d) Prove or give a counterexample: An edge is crucial if and only if it is contained in all minimum cuts. **Answer:** False, BT above is a counterexample: it is a crucial edge that does not belong to cut (SA,SB).

Problem 4 (20 points)

True or False? Circle the right answer. No justification is needed. No points will be subtracted for wrong answers, so it is to your best interest to guess all you want.

T The shortest paths between all pairs of points in a weighted graph with n nodes (and no negative cycles) can be found in time $O(n^3)$.

not graded If the graph is sparse with $|E| = O(n \log n)$, a faster than $\Theta(n^3)$ algorithm is possible for the previous question.

F The solution of $T(n) = T(n-1) + n^2$ is $\Theta(n^4)$.

T In Huffman's algorithm with all frequencies different, the symbol with the second-lowest frequency is always guaranteed to be furthest (or tied for furthest) from the root.

T In Huffman's algorithm with all frequencies different, the symbol with the highest frequency is always guaranteed to be closest (or tied for closest) to the root.

T The dynamic programming algorithm for the Traveling Salesman Problem runs in exponential time.

F Simplex runs in polynomial time.

T There is a polynomial algorithm for solving linear programming.

T The Linear Programming problem is in NP.

T If problem A is in P and problem B is NP-complete, then A reduces to B.