×

Problem Set #6

Back to Week 6

4/5 points earned (80%)

Quiz passed!

✓ 1 / 1 points

1.
Which of the following statements is NOT true about the generic local search algorithm?

○ The output of the generic local search algorithm generally depends on the choice of the starting point.

[✓ The generic local search algorithm is guaranteed to eventually converge to an optimal solution.]

Correct Response
Only a *locally* optimal solution.

○ In some cases, the generic local search algorithm can require an exponential number of iterations before terminating.

○

The output of the generic local search algorithm generally depends on the choice of the superior neighboring solution to move to next (in an iteration where there are multiple such solutions).

✗ 0 / 1 points

2.
Recall the Vertex Cover problem from the video lectures: given an undirected graph (with no parallel edges), compute a minimize-size subset of vertices that includes at least one endpoint of every edge. Consider the following greedy algorithm, given a graph G: (1) initialize $S = \emptyset$; (2) while S is not a vertex cover of G: (2a) let F denote the edges with neither endpoint in S; (2b) let e be some edge of F; (2c) add both endpoints of e to S; (3) return S.

Consider the following statement: for every graph G with n vertices, this greedy algorithm returns a vertex cover with size at most $f(n)$ times that of an optimal (minimum-size) vertex cover. Which of the following is the smallest choice of the function $f(n)$ that makes this statement true?

[Hint: suppose the greedy algorithm picks an edge e with endpoints u and v. What can you say about every feasible solution to the problem?]

- ○ $O(n)$
- [✓ $O(\sqrt{n})$]

Incorrect Response
The key to understanding the algorithm's performance is the set M of edges chosen in step 2b (one edge per iteration of the main while loop). Do you see why every vertex cover has size at least $|M|$?

- ○ 2
- ○ $O(\log n)$

✓ 1 / 1 points

3.

Consider the following job scheduling problem. There are m machines, all identical. There are n jobs, and job j has size p_j. Each job must be assigned to exactly one machine. The *load* of a machine is the sum of the sizes of the jobs that get assigned to it. The *makespan* of an assignment of jobs is the maximum load of a machine; this is the quantity that we want to minimize. For example, suppose there are two machines and 4 jobs with sizes 7,8,5,6. Assigning the first two jobs to the first machine and the last two jobs to the second machine yields machine loads 15 and 11, for a makespan of 15. A better assignment puts the first and last jobs on the first machine and the second and third jobs on the second machine, for a makespan of 13.

Consider the following greedy algorithm. Iterate through the jobs $j = 1, 2, 3, \ldots, n$ one-by-one. When considering job j, assign it to the machine that currently has the smallest load (breaking ties arbitrarily). For example, in the four-job instance above, this algorithm would assign the first job to the first machine, the second job to the second machine, the third job to the first machine, and the fourth job to the second machine (for a suboptimal makespan of 14).

Consider the following statement: for every such job scheduling instance, this greedy algorithm computes a job assignment with makespan at most c times that of an optimal (minimum-makespan) job assignment. Which of the following is the smallest choice of the constant c that makes this statement true?

[Hint: let A and B denote the average and maximum job sizes ($A = (\sum_j p_j)/m$ and $B = \max_j p_j$). Try to relate both the optimal solution and the output of the greedy algorithm to A, B.]

[✓ 2]

Correct Response
For the upper bound, let A and B denote the total and maximum job sizes ($A = \sum_j p_j$ and $B = \max_j p_j$). Certainly $OPT \geq B$. Prove that $OPT \geq A/m$, as well (with equality only if OPT spreads out the jobs perfectly). By considering the iteration that schedules the job that determines the makespan, prove that the makespan of the greedy algorithm is at most $B + A/m$.

4

6/5

3/2

✓ 1 / 1 points

4.
Consider the same makespan-minimization job scheduling problem studied in the previous problem. Now suppose that, prior to running the greedy algorithm in the previous problem, we first *sort* the jobs from biggest to smallest. For example, in the four-job instance discussed in the previous problem, the jobs would be considered in the order 8,7,6,5, and the greedy algorithm would then produce an optimal schedule, with makespan 13.

Consider the following statement: for every such job scheduling instance, the greedy algorithm (with this sorting preprocessing step) computes a job assignment with makespan at most c times that of an optimal (minimum-makespan) job assignment. Which of the following is the smallest choice of the constant c that makes this statement true?

- ○ 6/5
- [✓ 3/2]

Correct Response
We refine the previous solution by replacing the lower bound B (previously the max job size) with a different quantity. Consider the job that determines the makespan. The interesting case is where this job j is not one of the first m jobs (check this!). By the greedy ordering, then, $p_j \leq p_{m+1}$. Since every assignment must place two of the first $m + 1$ jobs on a common machine, OPT is at least $2p_{m+1}$ and hence p_j is at most OPT/2. Since the load of j's machine was at most OPT before it was assigned (as in the previous problem), the bound follows. Optional: can you prove a bound that is even better than $3/2$?

- ○ 4
- ○ 2

✓ 1 / 1 points

5.

Suppose we apply local search to the *minimum* cut problem. Given an undirected graph, we begin with an arbitrary cut (A, B). We check if there is a vertex v such that switching v from one group to the other would strictly decrease the number of edges crossing the cut. (Also, we disallow vertex switches that would cause A or B to become empty.) If there is such a vertex, we switch it from one group to the other; if there are many such vertices, we pick one arbitrarily to switch. If there are no such vertices, then we return the current locally optimal cut (A, B). Which of the following statements is true about this local search algorithm?

- ○ If this local search algorithm is guaranteed to terminate in a polynomial number of iterations, it would immediately imply P=NP.
- [✓ This local search algorithm is guaranteed to terminate in a polynomial number of iterations.]

Correct Response

Every iteration strictly decreases the number of crossing edges, so there can be only m iterations.

- ○ This local search algorithm is guaranteed to compute a minimum cut.
- ○ This local search algorithm is guaranteed to compute a cut for which the number of crossing edges is at most twice the minimum possible.