¡Felicitaciones! ¡Aprobaste!

Calificación recibida 100 % Para Aprobar 80 % o más

Ir al siguiente elemento

Problem Set #3

Calificación de la entrega más reciente: 100 %

1. 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.

1 / 1 punto

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 $f(n)$ for which this statement is 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?]

- (●) 2
- ( ) $O(\log n)$
- ( ) $O(\sqrt{n})$
- ( ) $O(n)$

Correcto

Let M be the edges chosen in step 2b (one edge per iteration of the main loop). M is a matching --- that is, these edges are pairwise disjoint. Thus, every vertex cover has size at least $|M|$. The size of the output of the greedy algorithm is exactly $2|M|$.

2. In the *set cover* problem, you are given m sets $S_1, ..., S_m$, each a subset of a common set U with size $|U| = n$. The goal is to select as few of these sets as possible, subject to the constraint that the union of the chosen sets is all of U. (You can assume that $\cup_{i=1}^{m} S_i = U$.) For example, if the given sets are $\{1, 2\}$, $\{2, 3\}$, and $\{3, 4\}$, then the optimal solution consists of the sets $\{1, 2\}$ and $\{3, 4\}$.

1 / 1 punto

Here is a natural iterative greedy algorithm. First, initialize $C = \emptyset$, where C denotes the sets chosen so far. The main loop is: as long as the union $\cup_{S \in C} S$ of the sets chosen so far is not the entire set U:

- Let S_i be a set that includes the maximum-possible number of elements not in previously-chosen sets (i.e., that maximizes $|S_i - \cup_{S \in C} S|$).
- Add S_i to C.

Consider the following statement: for every instance of the set cover problem (with $|U| = n$), this greedy algorithm returns a set cover with size at most $f(n)$ times that of an optimal (minimum-size) set cover. Which of the following is the smallest choice of the function $f(n)$ for which this statement is true?

[Hint: what's the mininum-possible progress that the greedy algorithm can make in each iteration, as a function of the size of an optimal set cover and of the number of elements that have not yet been covered?]

- ( ) 2
- (●) $O(\log n)$
- ( ) $O(\sqrt{n})$
- ( ) $O(n)$

Correcto

Let OPT denote the minimum size of a set cover. The key observation is: in a given iteration of the greedy algorithm, if there are still x elements not covered by the sets chosen thus far, then the next set chosen will cover at least x/OPT new elements (why?).

The key observation implies that the greedy algorithm, after choosing OPT sets, will have covered at least a constant fraction of U. (Since $(1 - 1/OPT)^{OPT}$ is roughly $1/e$, where $e = 2.718....$) Thus after $O(\log n)$ phases of choosing OPT sets, the greedy solution must cover all of U.

3. Suppose you are given m sets $S_1, ..., S_m$, each a subset of a common set U. The goal is to choose 2 of the m sets, S_i and S_j, to maximize the size $|S_i \cup S_j|$ of their union. One natural heuristic is to use a greedy algorithm: (i) choose the first set S_i to be as large as possible, breaking ties arbitrarily; (ii) choose the second set S_j to maximize

1 / 1 punto

$|S_i \cup S_j|$ (i.e., as the set that includes as many elements as possible that are not already in S_i), again breaking ties arbitrarily. For example, if the given sets are $\{1,2\}$, $\{2,3\}$, and $\{3,4\}$, then the algorithm might pick the set $\{1,2\}$ in the first step; if it does so, it definitely picks the set $\{3,4\}$ in the second step (for an objective function value of 4).

Consider the following statement: for every instance of the above problem, the greedy algorithm above chooses two sets S_i, S_j such that $|S_i \cup S_j|$ is at least c times the maximum size of the union of two of the given sets. Which of the following is the largest choice of the constant c for which this statement is true?

- (●) $3/4$
- ( ) $1/2$
- ( ) 1
- ( ) $2/3$

Correcto

Let OPT denote the maximum size of the union of two of the given sets. The first set S_i chosen by the greedy algorithm has size at least $OPT/2$ (why?). Let x denote the number of elements in an optimal solution not already included in S_i. Then the second set S_j chosen by the greedy algorithm includes at least $x/2$ elements not covered already by S_i (why?). Taken together, the two sets then include at least $OPT * (3/4)$ elements (why?).

The example in the problem description already shows that no constant better than $3/4$ is possible (if the greedy algorithm chooses the set $\{2,3\}$ in the first iteration, then its solution covers only 3 elements, while the optimal solution covers all 4).

4. 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. **1 / 1 punto**

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
- ( ) $6/5$
- ( ) $3/2$
- ( ) 4

Correcto

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 $B + A/m$.

5. 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. **1 / 1 punto**

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 for which this statement is true?

- (●) $3/2$
- ( ) 2
- ( ) $6/5$
- ( ) 4

Correcto

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 $3/2$?