Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. They are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

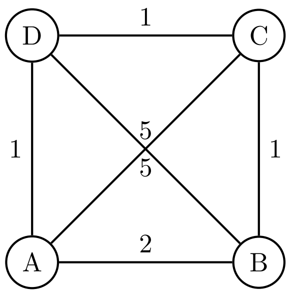
1 Approximating the Traveling Salesperson Problem

Recall in lecture, we learned the following approximation algorithm for TSP:

1. Given the complete graph $G = (V, E)$, compute its MST.
2. Run DFS on the MST computed in the previous step, and record down the the vertices visited in pre-order.
3. The output tour is the list of vertices computed in the previous step, with the first vertex appended to the end.

When G satisfies the triangle inequality, this algorithm achieves an approximation factor of 2. But what happens when triangle inequality does not hold?

Suppose we run this approximation algorithm on the following graph:



The algorithm will return different tours based on the choices it makes during its depth first traversal.

1. Which DFS traversal leads to the best possible output tour?

2. Which DFS traversal leads to the worst possible output tour?

3. What is the approximation ratio given by the algorithm in the worst case for the above instance? Why is it worse than 2?

Solution:

1. **A-D-C-B**, **D-C-B-A**, **B-C-D-A**, or **C-D-A-B**
   Explanation:
   The MST is {(A,D),(D,C),(C,B)}. The optimal tour uses all of these edges.
   For example, if we start traversing the tree at A, then the only traversal would be to follow the tree and go through the vertices in the order D, C, B
   Another potential traversal would be to start at D, move to C, then B, and backtrack, before going to A.
   Notice that if we started at D and moved to A first, then we would have to visit C next, and this would not lead to the best possible output tour (as per part (b)).
2. **C-B-D-A** or **D-A-C-B**
   See above explanation.
   Notice that the worst possible tour overall, **D-C-A-B**, is not possible to be output by this algorithm, regardless of the DFS traversal used.
3. $\frac{12}{5}$
   From previous parts, the optimal tour length is 5 while the worst possible is 12. This is worse than 2 because our graph does not satisfy the triangle inequality, specifically $\ell(A, C) > \ell(A, B) + \ell(B, C)$ and $\ell(B, D) > \ell(A, B) + \ell(D, A)$, which is the necessary condition for our algorithm to guarantee an approximation ratio of 2.

2 Boba Shops

A rectangular city is divided into a grid of $m \times n$ blocks. You would like to set up boba shops so that for every block in the city, either there is a boba shop within the block or there is one in a neighboring block (assume there are up to 4 neighboring blocks for every block). It costs r_{ij} to rent space for a boba shop in block ij.

Write an integer linear program to determine on which blocks to set up the boba shops, so as to minimize the total rental costs.

(a) What are your variables, and what do they mean?

(b) What is the objective function? Briefly justify.

(c) What are the constraints? Briefly justify.

(d) Solving the non-integer version of the linear program yields a real-valued solution. How would you round the LP solution to obtain an integer solution to the problem? Describe the algorithm in at most two sentences.

(e) What is the approximation ratio of your algorithm in part (d)? Briefly justify.

Solution:

(a) There is a variable for every block x_{ij}, i.e., $\{x_{ij} \mid i \in \{1, \dots, m\}, j \in \{1, \dots, n\}\}$. This variable corresponds to whether we put a boba shop at that block or not.

(b) $\min \sum_{i=1}^{m} \sum_{j=1}^{n} r_{ij} x_{ij}$. Alternatively, $\min t$ is correct as well as long as the correct constraint is added.

(c) (i) $x_{ij} \geq 0$: This constraint just corresponds to saying that there either is or isn't a boba shop at any block. $x_{ij} \in \{0, 1\}$ or $x_{ij} \in \mathbb{Z}_+$ is also correct.

(ii) For every $1 \leq i \leq m, 1 \leq j \leq n$:

$$x_{ij} + x_{(i+1),j} \mathbb{1}_{\{i+1 \leq m\}} + x_{(i-1),j} \mathbb{1}_{\{i-1 \geq 1\}} + x_{i,(j+1)} \mathbb{1}_{\{j+1 \leq n\}} + x_{i,(j-1)} \mathbb{1}_{\{j-1 \geq 1\}} \geq 1$$

This constraint corresponds to that for every block, there needs to be a boba shop at that block or a neighboring block.

$\mathbb{1}_{\{i+1 \leq m\}}$ means "1 if $\{i + 1 \leq m\}$, and 0 otherwise". It keeps track of the fact that we may not have all 4 neighbors on the edges, for instance.

(iii) If the objective was $\min t$, then the constraint $\sum_{i=1}^{m} \sum_{j=1}^{n} r_{ij} x_{ij} \leq t$ needs to be added.

(d) Round to 1 all variables which are greater than or equal to 1/5. Otherwise, round to 0.

In other words, put a boba shop on (i, j) iff $x_{i,j} \geq 0.2$.

(e) Using the rounding scheme in the previous part gives a 5-approximation.

Notice that every constraint has at most 5 variables. So for every constraint, there exists at least one variable in the constraint which has value $\geq 1/5$ (not everyone is below average). The total cost of the rounded solution is at most $5 \cdot$ LP-OPT, since $r_{ij} \leq 5 r_{ij} x_{ij}$ for any x_{ij} that gets rounded up, and the other i, j pairs contribute nothing to the cost of the rounded solution. Since Integral-OPT $\geq$ LP-OPT (the LP is more general than the ILP), our rounding gives value at most 5 LP-OPT $\leq$ 5 Integral-OPT. So we get a 5-approximation.

3 Maximum Coverage

In the maximum coverage problem, we have m subsets of the set $\{1, 2, \ldots n\}$, denoted $S_1, S_2, \ldots S_m$. We are given an integer k, and we want to choose k sets whose union is as large as possible.

Give an efficient algorithm that finds k sets whose union has size at least $(1 - 1/e) \cdot OPT$, where OPT is the maximum number of elements in the union of any k sets. In other words, $OPT = \max_{i_1, i_2, \ldots i_k} |\cup_{j=1}^{k} S_{i_j}|$. Provide an algorithm description and justify the lower bound on the number of elements covered by your solution.

Hint: be greedy! For the proof, you may use the follow property without proof: $(1 - 1/n)^n \le 1/e$ *for all* $n \in \mathbb{Z}$.

Solution: Similar to set cover, we choose sets greedily, each time adding the set that includes the most elements that are not covered by any sets included in our solution so far.

Let n_i be OPT minus the number of elements covered by the first i sets we choose. Initially, we have $n_0 = OPT - 0 = OPT$. Note that for any i, the k sets forming OPT cover at least n_i elements that our solution does not cover, which means one of these sets covers at least n_i/k elements our solution doesn't cover. The set we add in iteration $i + 1$ can is guaranteed to cover more new elements than this set, i.e. the $(i + 1)$-th set we add covers at least n_i/k new elements. Thus, we have

$$\begin{aligned} n_{i+1} &= OPT - (\text{number of elements covered by the first } i+1 \text{ sets}) \\ &= OPT - (OPT - n_i + (\text{number of new elements covered by the } (i+1)\text{-th set})) \\ &\le OPT - (OPT - n_i + n_i/k) \\ &= n_i - n_i/k \\ &= n_i(1 - 1/k) \end{aligned}$$

In turn, we have $n_k \le (1 - 1/k)^k n_0 = (1 - 1/k)^k OPT \le \frac{1}{e} \cdot OPT$, or equivalently our solution covers at least $(1 - 1/e) \cdot OPT$ elements.