CMPS 6610 Problem Set 05

In this assignment we'll look at the shortest paths and spanning trees.

To make grading easier, please place all written solutions directly in `answers.md`, rather than scanning in handwritten work or editing this file.

All coding portions should go in `main.py` as usual.

1. Shortest shortest paths

a) Suppose we are given a a directed, **weighted** graph $G = (V, E)$ with only positive edge weights. For a source vertex s, design an algorithm to find the shortest path from s to all other vertices with the fewest number of edges. That is, if there are multiple paths with the same total edge weight, output the one with the fewest number of edges.

put in answers.md

.
.
.

b) What is the work and span of your algorithm?

put in answers.md

.
.
.

2. Expression Trees

Let's consider a classic application of trees for representing arithmetic expressions. Suppose that we have a given set of associative binary operators and are given a parenthesized arithmetic expression with n literals and operators, where a literal is a variable or a constant (e.g., `((a+b)+c)*(-3))`). Any such expression can also be represented as a binary tree in which operators are internal nodes and literals are leaves.

a) Design an algorithm that takes an expression E with n literals and produces a tree T_E. Analyze and state the work and span of your algorithm.

put in answers.md

.
.
.

b) Now, suppose we are given a tree T_E and values for each variable as input. What is the work and span to compute the value of the corresponding arithmetic expression?

put in answers.md

.
.
.

c) Finally, suppose we have evaluated an expression as described above but now want to change just one variable's value. What is the work/span to update a previously computed value, with respect to the expression tree?

put in answers.md

.
.
.

3. Improving Dijkstra

In our analysis of the work done by Dijkstra's algorithm, we ended up with a bound of $O(|E| \log |E|)$. Let's take a closer look at how changing the type of heap used affects this work bound.

a) A d-ary heap is a generalization of a binary heap in which we have a d-ary tree as the data structure. The heap and shape properties are still maintained, but each internal node now has d children (except possibly the rightmost leaf). What is the maximum depth of a d-ary heap?

put in answers.md

.
.
.

b) In a binary heap the `delete-min` operation removes the root, places the rightmost leaf at the root position and restores the heap property by swapping downward. Similarly the `insert` operation places the new element as the rightmost leaf and swaps upward to restore the heap property. What is the work done by `delete-min` and `insert` operations in a d-ary heap? Note that the work differs for each operation.

put in answers.md

.
.
.

c) Now, suppose we use a d-ary heap for Dijkstra's algorithm. What is the new bound on the work? Your bound will be a function of $|V|$, $|E|$, and d and will account for the `delete-min` and `insert` operations separately.

put in answers.md

.
.
.

d) Now that we have a characterization of how Dijkstra's algorithm performs with a d-ary heap, let's look at how we might be able to optimize the choice of d under certain assumptions. Let's suppose that we have a moderate number of edges, that is $|E| = |V|^{1+\epsilon}$ for $0 < \epsilon < 1$. What value of d yields an overall running time of $O(|E|)$?

put in answers.md

.
.
.

4. All-pairs shortest paths using Dynamic Programming

Suppose we wanted to compute the shortest paths between all pairs of vertices. Dijkstra's algorithm focuses on a single source, so a natural question is whether we can do better than simply running Dijkstra n times. Let's consider a dynamic programming approach.

Suppose that we label the vertices from 0 to $n-1$, and let $APSP(i, j, k)$ be the weight of the shortest path between vertices i and j such that only vertices $0, 1, \ldots, k$ are allowed to be used. Then the cost of the shortest path between vertices i and j is then given by $APSP(i, j, n-1)$.

a) Consider the following graph with 3 vertices.

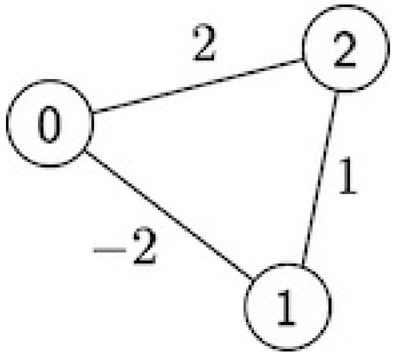

Figure 1: apsp_example.jpg

Compute $APSP(i, j, k)$ for all i, j, k.

put in answers.md

.

.

.

b) Do you see a relationship between $APSP(i, j, 1)$ and $APSP(i, j, 2)$? Can you write $APSP(i, j, 2)$ in terms of $APSP(i, j, 0)$ and $APSP(i, j, 1)$ only?

put in answers.md

.

.

.

c) Suppose that an oracle (i.e., an all-powerful source of information) makes available to us all possible values for $APSP(i, j, k - 1)$ for all i, j and some particular value of $k - 1 < n$. Then what is the shortest path cost $APSP(i, j, k)$? Well, it is either $APSP(i, j, k - 1)$, or some other path from i to j that has length k. Generalize your observation from b) above to give an optimal substructure property for $APSP(i, j, k)$.

put in answers.md

.

.

.

d) As usual naively implementing this optimal substructure property to compute $APSP(i, j, n - 1)$ for all i, j will be inefficient. Suppose we perform top-down memoization so that we only ever compute each subproblem from scratch once. How many distinct subproblems will be computed from scratch, and what is the resulting work of this dynamic programming algorithm?

put in answers.md

.

.

.

e) Compare the work of this algorithm against that of just running Dijkstra n times. Is our dynamic programming algorithm better than this?

put in answers.md

.
.
.

5. Spanning trees

a) Consider a variation of the MST problem that instead asks for a tree that minimizes the maximum weight of any edge in the spanning tree. Let's call this the minimum maximum edge tree (MMET). Is a solution to MST guaranteed to be a solution to MMET? Why or why not?

put in answers.md

.
.
.

b) Suppose that the optimal solution to MST is impossible to use for some reason. Describe an algorithm to instead find the next best tree (pseudo-code or English is fine). That is, return the tree with the next lowest weight.

put in answers.md

.
.
.

c) What is the work of your algorithm?

put in answers.md

.
.
.