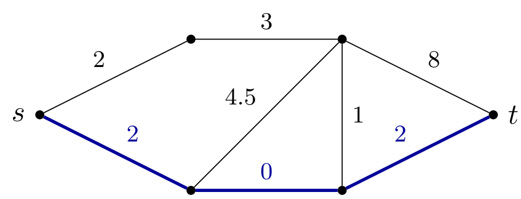
Math 416

Worksheet 15. Dijkstra's Algorithm

Minimum-weight paths Now our input will be a **weighted graph**, i.e., a graph in which each edge e comes with a **weight** or **cost** $\mathrm{wt}(e) \geq 0$. The weight of a path is defined to be the sum of weights of edges on the path.

Given vertices s and t in the graph, we want to find a minimal-weight path from s to t, as in this picture.



If we knew the k vertices $v_1, \ldots, v_k$ closest to s and their distances to s, then we could find the $(k+1)^{\text{st}}$-closest vertex v_{k+1} as follows. Vertices on a minimum-weight path from v_{k+1} to s must be closer to s, so among $v_1, \ldots, v_k$. Examine all the 'frontier edges' (v_j, w) and find such a w that minimizes the distance from s to v_j plus the weight of the edge (v_j, w). This w will be v_{k+1}.

Problem 1. Fill in the gaps in the pseudocode for Dijkstra's algorithm, below.

Algorithm 1: Dijkstra's algorithm

Input: a weighted graph $G = (V, E, \mathrm{wt})$ (with nonnegative weights) and a vertex $s \in V$
Output: an array dist so that $\mathrm{dist}(v)$ is the length of a min-weight path from s to v

```
set dist(s) = 0;
set S = {s} ;                                          // S is the set of marked or explored nodes
foreach v ≠ s do
    set dist(v) = ∞;
while S ≠ V do
    find w ∉ S a vertex adjacent to a v ∈ S for which [                    ] ;
    record pred(w) = v ;                               // for later analysis
    add w to S ;
    set dist(w) = [            ] ;
end
```

Problem 2. After running the algorithm, how can you find the min-weight path from s to your favorite vertex t? (*Hint:* use pred)

Remarks. (i) Dijkstra's algorithm works just as well for weighted digraphs.

(ii) Dijkstra's algorithm is 'greedy' in the sense that we always extend a path by adding a single edge minimizing some objective function.

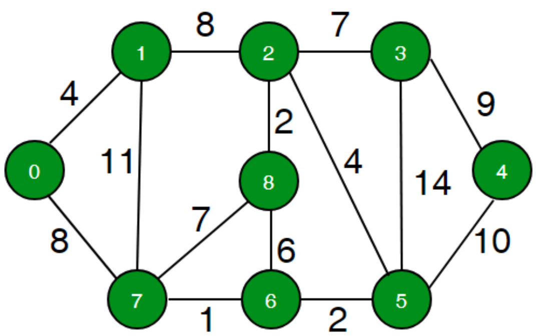
(iii) From the right point of view, Dijkstra's algorithm generalizes breadth-first search.

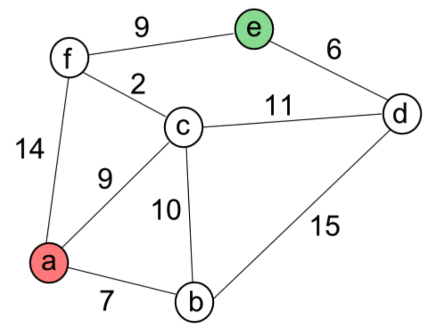
(iv) Dijkstra's algorithm no longer works if weights are allowed to be negative, as you'll see.

(v) The `while` loop iterates $\leq n-1$ times, where $n = |V|$. (Why?) It seems like choosing the right w in the loop will require a search over $V \setminus S$ and for each $w \in V \setminus S$ a computation of $\min_{v \in S} \mathrm{dist}(v) + \mathrm{wt}(v, w)$, which in the end looks like $O(mn)$ time. (Here m is the number of edges.) The algorithm can be implemented more carefully to achieve $O(m \log n)$ time.

Problem 3. Dijkstra's algorithm produces a tree of minimum-weight paths, if it is run on a connected graph. Describe the tree precisely (i.e., give an exact description of which edges are in the tree, in terms of variables in the algorithm) and explain why it is a tree.

Problem 4. By executing Dijkstra's algorithm, find minimal weight paths from (0)-(4) and (a)-(e) respectively in the below graphs. Plot the corresponding minimal weight trees.





Problem 5. What does Dijkstra's algorithm do on an unweighted graph? That is, suppose that all the weights in a graph are 1 and describe the execution of Dijkstra's algorithm.

Problem 6. Spend two minutes (i.e. don't spend too long) with your groupmates speculating about how Dijkstra's algorithm can be implemented to run in $O(m \log n)$ time. (*Hint:* Explicitly maintain values of $\min_{v \in S} \operatorname{dist}(v) + \operatorname{wt}(v, w)$ for all $w \notin S$ rather than recomputing them in each iteration. Keep the nodes of $V \setminus S$ in a 'priority queue' with the values $\min_{v \in S} \operatorname{dist}(v) + \operatorname{wt}(v, w)$ as keys. Look up the definition of a priority queue if you need to! This is actually a bit subtle: see the discussion on pp. 661–662 of CLRS, if you are interested in understanding the whole argument.)

Correctness of Dijkstra's Algorithm. We are proving the validity of a certain loop invariant for Dijkstra.

Problem 7. What is the loop invariant?

Claim. Consider the set S during the execution of Dijkstra's algorithm, at the start of the `while` loop on line 5 of the algorithm.

(1) For all $u \in S$, the quantity $\operatorname{dist}(u)$ is the length of the minimum-weight path from s to u; and
(2) P_u is a path of minimum weight from u to s. (By P_u we mean the path obtained by starting at u and iterating pred until you reach s.)

Problem 8. Fill in the following outline of a proof of the Claim.

(a) We proceed by induction on $|S|$. Verify that the Claim holds for $|S| = 1$.
(b) Now suppose that the claim holds for $|S| = k$. Suppose that w is discovered in line 6 of the algorithm to make $|S| = k + 1$, say with $\operatorname{pred}(w) = v$. By induction, P_v is a min-weight path from s to v. How does P_w relate to P_v?
(c) Consider any path P from s to w. We must show that $\operatorname{wt}(P) \geq \operatorname{wt}(P_w)$. Since $s \in S$ but $w \notin S$, there is a first vertex y on P that does not belong to S. Let x be the previous vertex on P, so that $x \in S$. Let P' be the portion of P from s to x. What can you say about $\operatorname{wt}(P')$ versus $\operatorname{wt}(P_x)$?
(d) (Draw a picture and) Complete the proof that $\operatorname{wt}(P) \geq \operatorname{wt}(P_w)$.
(e) Verify that you have a complete proof of the Claim.
(f) At what point (if any) does your proof use that the algorithm chose w instead of y?
(g) At what point (if any) does your proof use that weights are nonnegative?