16 June

Dijkstra's algorithm. [P&S §6.4]

Streamlined version of primal-dual for shortest path (for nonnegative arc weights).

Slight modifications:

- W will grow forward from s rather than backward from t. (Equivalent to reversing the directions of all arcs and finding a shortest path from t to s using primal-dual.)
- We will find shortest distances from s to every other node.
- We will maintain, in every iteration, a label $\rho(x)$ for each node $x \in V$ such that

$\rho(x) =$ shortest length of any path from s to x passing through only nodes in W, or ∞ if no such path exists

and use $\rho(x)$ to guide the algorithm.

- Must have nonnegative edge weights c_{ij}.
- For ease of notation, take $c_{ij} = \infty$ if the arc (i,j) does not exist in the graph.

Dijkstra's algorithm — 2

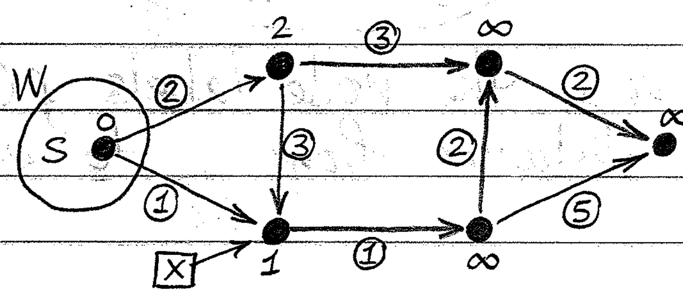
16 June

Outline of Dijkstra's algorithm

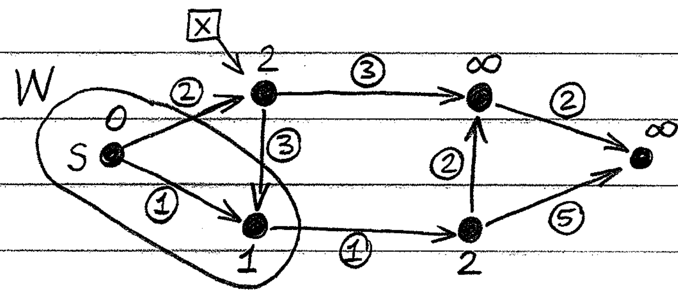
1. Initialize $W := \{s\}$, $\rho(s) := 0$, $\rho(i) := c_{si}$ for all $i \neq s$.
2. If $W = V$ (all vertices are in W), we are done.
3. Find $\min\{\rho(y) : y \notin W\}$, say $\rho[x]$.
4. Add x to W.
5. For all $y \notin W$, set $\rho(y) := \min\{\rho(y), \rho(x) + c_{xy}\}$.
6. Go to step 2.

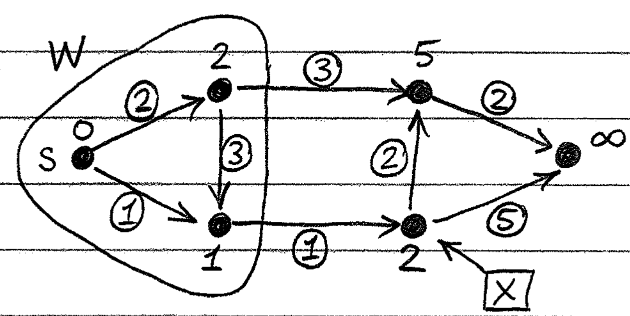
Because the new shortest distance from s to y through W is either the old shortest distance from or the shortest distance from s to x plus the distance from x to y.

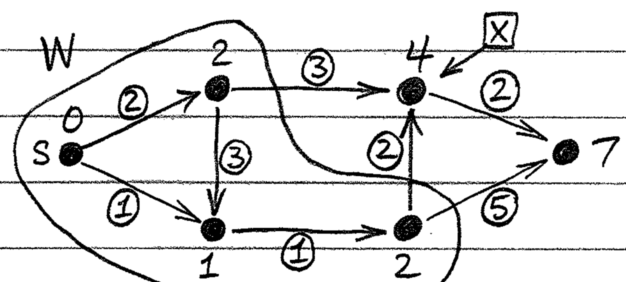
Example.

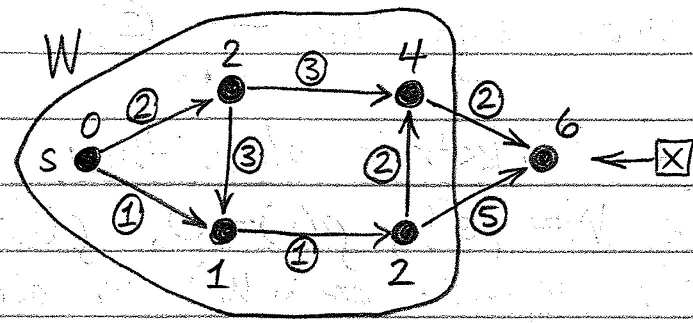


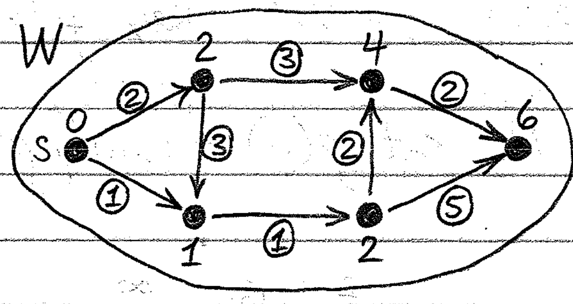
Arc weights are circled. $\rho(i)$ written next to nodes.





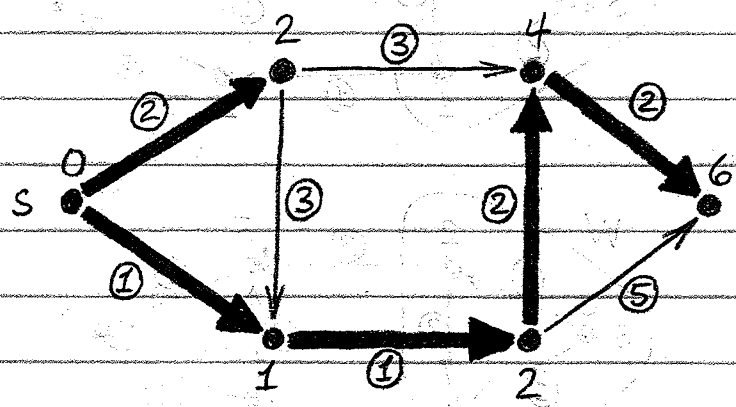






At the end, the node labels $\rho(i)$ give the shortest distances from s to each node.

To get shortest paths, use the admissible arcs: those arcs (i,j) for which $\rho(j) - \rho(i) = c_{ij}$.



16 June

Some definitions from graph theory. [P&S §A.2, pp. 20–23]

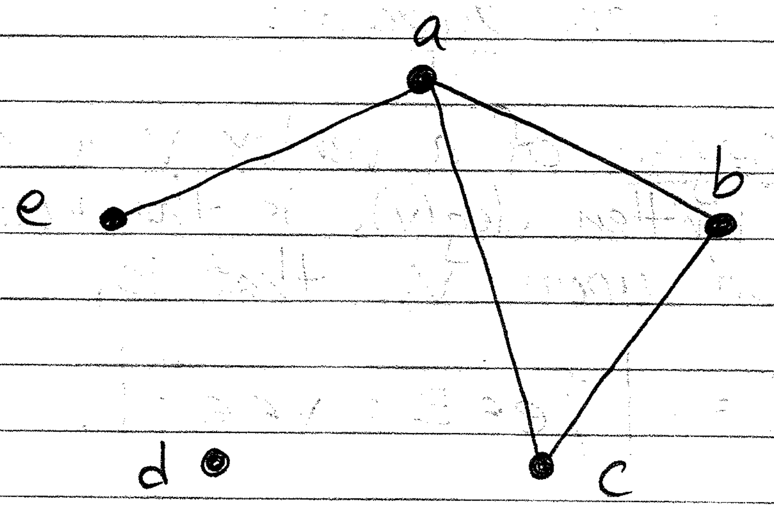
Defn. A (simple, undirected) graph G is an ordered pair $G=(V,E)$, where V is a nonempty set called the vertex set, and E is a set called the edge set whose elements (the edges of G) are subsets of V of cardinality 2.

— Normally graphs are drawn with points to represent the vertices (the elements of V) and lines or curves joining those points to represent the edges.

Example. $G=(V,E)$, where

$$V=\{a,b,c,d,e\},$$

$$E=\{\{a,b\},\{a,c\},\{a,e\},\{b,c\}\}.$$



Note that the geometry of the picture is not important. All that matters is which points are joined to which others.

Defn. The cardinality of V, $|V|$, is called the order of the graph. The cardinality of E, $|E|$, is called the size of the graph.

— Often the letter n is used for the order (the number of vertices), and the letter m is used for the size (the number of edges).

Defn. If $e = \{u, v\} \in E$ is an edge in a graph, then we say:

- u and v are the endpoints of e;
- e is incident upon u (and upon v);
- u and v are adjacent vertices;
- v is a neighbor of u, and vice versa;
- e joins u and v.

Defn. If $e, f \in E$ are two distinct edges and $e \cap f \neq \emptyset$ (i.e., e and f have an endpoint in common), then we say that e and f are adjacent.

Defn. The degree of a vertex v in a graph $G = (V, E)$, written $\deg(v)$, is the number of edges incident upon v, that is,

$$\deg(v) = |\{e \in E : v \in e\}|.$$

Graph definitions — 2

16 June

Defn. A *walk* in a graph $G=(V,E)$ is a sequence $(v_0, v_1, v_2, \ldots, v_k)$ such that $v_i \in V$ for all $0 \le i \le k$ and $\{v_i, v_{i+1}\} \in E$ for all $0 \le i \le k-1$. The *length* of this walk is k.

Defn. A *path* in a graph $G=(V,E)$ is a walk $(v_0, v_1, v_2, \ldots, v_k)$ such that $v_0, v_1, v_2, \ldots, v_k$ are all *distinct* vertices.

Defn. A *cycle* in a graph $G=(V,E)$ is a walk $(v_0, v_1, v_2, \ldots, v_k)$ such that $k \ge 3$, $(v_0, v_1, \ldots, v_{k-1})$ is a path, and $v_k = v_0$.

Defn. A graph $G=(V,E)$ is *connected* if for all $u,v \in V$ there exists a path in G from u to v (that is, a path $(v_0, v_1, v_2, \ldots, v_k)$ such that $v_0 = u$ and $v_k = v$). If a graph is not connected, it is *disconnected*.

Defn. A graph is *cyclic* if it contains a cycle. Otherwise it is *acyclic*.

Defn. A *tree* is a connected, acyclic graph.
— P&S like to use T for the *edge set* of a tree.

Defn. A *forest* is an acyclic graph.
— So a tree is a connected forest.

Defn. A *leaf* of a tree is a vertex of degree 1.

Defn. A subgraph of a graph $G=(V,E)$ is a graph $H=(V',E')$ such that $V' \subseteq V$, $E' \subseteq E$, [and $e' \subseteq V'$ for all $e' \in E'$ (i.e., every edge in E' joins two vertices in V').]

— The last part in square brackets is not strictly necessary to say, because it is implied by the fact that (V',E') is a graph.

Defn. A subgraph $H=(V',E')$ of a graph $G=(V,E)$ is spanning if $V'=V$.

Defn. A spanning tree of a graph is a spanning subgraph that is a tree.

Defn. A subgraph $H=(V',E')$ of a graph $G=(V,E)$ is a connected component of G if H is maximally connected, i.e., H is connected but every proper supergraph of H [every subgraph $K=(V'',E'')$ of G such that $V'' \supseteq V'$ and $E'' \supseteq E'$ but $H \neq K$] is disconnected.

[Maximally under the subgraph relation.]

— Observe: Every connected component of a forest is a tree. (That's why it's called a forest.)

Another way to say this: A subgraph H of a graph G is a connected component of G if and only if H is connected and the only connected subgraph of G that contains H as a subgraph is H itself.