

Analysis of Algorithms

Homework 3

Abraham Murciano

November 30, 2020

1 The Kruskal and Prim Algorithms

We are given the following weighted, undirected graph $G = (V, E)$ where V is the set of vertices, E is a set of edges, and $W_{u,v}$ is the weight from vertices u to v .

$$\begin{aligned} V &= \{a, b, c, d, e, f, g, h, i\} \\ E &= \{\{a, b\}, \{a, h\}, \{a, i\}, \{b, c\}, \{b, f\}, \{c, d\}, \{d, e\}, \\ &\quad \{d, g\}, \{d, h\}, \{e, f\}, \{f, g\}, \{h, i\}\} \\ W_{a,b} &= 4, W_{a,h} = 10, W_{a,i} = 6, W_{b,c} = 7, W_{b,f} = 12, W_{c,d} = 8, W_{d,e} = 3, \\ W_{d,g} &= 5, W_{d,h} = 2, W_{e,f} = 11, W_{f,g} = 1, W_{h,i} = 9 \end{aligned}$$

1.1 Kruskal

We are to run the Kruskal algorithm on this graph, showing intermediate stages. We begin with the set of edges to return F , initialised to ϕ . We also begin with a disjoint set D , starting with each vertex in its own set; i.e $D = \{\{v\} : v \in V\}$.

The set the algorithm returns as a minimum spanning forrest is

$$F = \{\{a, b\}, \{a, i\}, \{b, c\}, \{c, d\}, \{d, e\}, \{d, g\}, \{d, h\}, \{f, g\}\}$$

as shown by the intermediate steps in table 1.

Prim

Using the same example we are to run Prim's algorithm which returns the same thing. We start with an set of vertices V_0 initialised to an arbitrary vertex (here we choose a), which we insert vertices into one at a time. We also use a set E_0 which is the subset of edges to return, initialised to ϕ . By the end,

$$E_0 = \{\{a, b\}, \{a, i\}, \{b, c\}, \{c, d\}, \{d, e\}, \{d, g\}, \{d, h\}, \{f, g\}\}$$

Step	Variable	Value
1	F	$\{\{f, g\}\}$
	D	$\{\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f, g\}, \{h\}, \{i\}\}$
2	F	$\{\{d, h\}, \{f, g\}\}$
	D	$\{\{a\}, \{b\}, \{c\}, \{d, h\}, \{e\}, \{f, g\}, \{i\}\}$
3	F	$\{\{d, e\}, \{d, h\}, \{f, g\}\}$
	D	$\{\{a\}, \{b\}, \{c\}, \{d, e, h\}, \{f, g\}, \{i\}\}$
4	F	$\{\{a, b\}, \{d, e\}, \{d, h\}, \{f, g\}\}$
	D	$\{\{a, b\}, \{c\}, \{d, e, h\}, \{f, g\}, \{i\}\}$
5	F	$\{\{a, b\}, \{d, e\}, \{d, g\}, \{d, h\}, \{f, g\}\}$
	D	$\{\{a, b\}, \{c\}, \{d, e, f, g, h\}, \{i\}\}$
6	F	$\{\{a, b\}, \{a, i\}, \{d, e\}, \{d, g\}, \{d, h\}, \{f, g\}\}$
	D	$\{\{a, b, i\}, \{c\}, \{d, e, f, g, h\}\}$
7	F	$\{\{a, b\}, \{a, i\}, \{b, c\}, \{d, e\}, \{d, g\}, \{d, h\}, \{f, g\}\}$
	D	$\{\{a, b, c, i\}, \{d, e, f, g, h\}\}$
8	F	$\{\{a, b\}, \{a, i\}, \{b, c\}, \{c, d\}, \{d, e\}, \{d, g\}, \{d, h\}, \{f, g\}\}$
	D	$\{\{a, b, c, d, e, f, g, h, i\}\}$

Table 1: The steps taken during the execution of the Kruskal algorithm

Step	Variable	Value
1	V_0	$\{a\}$
	E_0	ϕ
2	V_0	$\{a, b\}$
	E_0	$\{\{a, b\}\}$
3	V_0	$\{a, b, i\}$
	E_0	$\{\{a, b\}, \{a, i\}\}$
4	V_0	$\{a, b, c, i\}$
	E_0	$\{\{a, b\}, \{a, i\}\}, \{b, c\}$
5	V_0	$\{a, b, c, d, i\}$
	E_0	$\{\{a, b\}, \{a, i\}\}, \{b, c\}, \{c, d\}$
6	V_0	$\{a, b, c, d, h, i\}$
	E_0	$\{\{a, b\}, \{a, i\}\}, \{b, c\}, \{c, d\}, \{d, h\}$
7	V_0	$\{a, b, c, d, e, h, i\}$
	E_0	$\{\{a, b\}, \{a, i\}\}, \{b, c\}, \{c, d\}, \{d, e\}, \{d, h\}$
8	V_0	$\{a, b, c, d, e, g, h, i\}$
	E_0	$\{\{a, b\}, \{a, i\}\}, \{b, c\}, \{c, d\}, \{d, e\}, \{d, g\}, \{d, h\}$
9	V_0	$\{a, b, c, d, e, f, g, h, i\}$
	E_0	$\{\{a, b\}, \{a, i\}, \{b, c\}, \{c, d\}, \{d, e\}, \{d, g\}, \{d, h\}, \{f, g\}\}$

Table 2: The steps taken during the execution of the Prim algorithm

2 Minimal Spanning Trees and Shortest Paths

It is not necessarily true that the path between any two vertices on a minimal spanning tree of a graph is also a shortest path between these two vertices on this graph. As a counter-example, suppose we have the following graph.

$$\begin{aligned}V &= \{a, b, c\} \\E &= \{\{a, b\}, \{b, c\}, \{a, c\}\} \\W_{a,b} &= 2, W_{b,c} = 2, W_{a,c} = 3\end{aligned}$$

Here, the shortest path between a and c is the single edge $\{a, c\}$ of weight 3, however the path between them in the minimal spanning tree is of weight 4.

3 Finding the Maximal Spanning Tree

We are to write an algorithm to find the maximal spanning tree of a graph. The algorithm presented below is a slight modification of Kruskal's algorithm, where instead of sorting the edges by increasing weight, we sort them by decreasing weight.

```
function MAXIMALSPANNINGTREE( $V, E$ )
   $F := \phi$ 
   $D := \text{DISJOINTSET}(V)$ 
  Sort  $E$  by decreasing weight
  for  $(u, v) \in E$  do
    if  $D.\text{FINDSET}(u) \neq D.\text{FINDSET}(v)$  then
       $F := F \cup \{(u, v)\}$ 
       $D.\text{UNION}(D.\text{FINDSET}(u), D.\text{FINDSET}(v))$ 
  return  $F$ 
```

The proof of its correctness is the same logic as that for Kruskal's algorithm, and its complexity is identical, namely $O(E\alpha(V))$, where α is the inverse Ackermann function. It uses $O(V)$ extra space because D uses $O(V)$ space, and the maximal spanning tree contains $|V| - 1$ edges.