SCC0602 - Algoritmos e Estruturas de Dados I

Minimum Spanning Trees

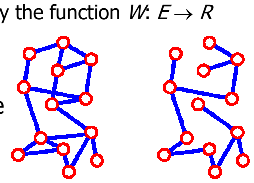
Professor: André C. P. L. F. de Carvalho, ICMC-USP
PAE: Rafael Martins D'Addio
Monitor: Joao Pedro Rodrigues Mattos

Today

- Weighted Graphs
- Minimum Spanning Trees
  - Greedy Choice Theorem
  - Kruskal Algorithm
  - Prim Algorithm

Spanning Tree

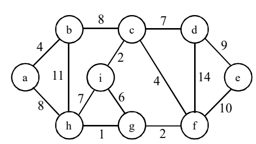
- Given an undirected, connected graph G = (V,E)
- A **spanning tree** of **G** is a subgraph which:
  - Contains all vertices of **G** (spans the graph G)
  - Each edge is weighted by the function *W*: $E \rightarrow R$
  - Is a tree
- How many edges are there in a spanning tree with *V* vertices?



G spanning tree of **G**

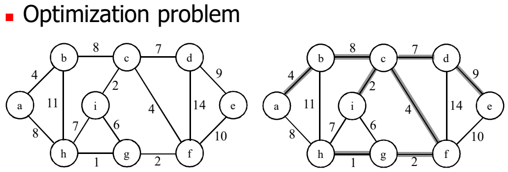
Minimum Spanning Trees

- **Minimum spanning tree** (MST)
  - Spanning tree *T* that connects all vertices minimizing total cost $w(T) = \sum_{(u,v) \in T} w(u,v)$
- How to find a MST?
  - Optimization problem



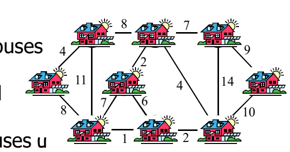
Minimum Spanning Trees

- **Minimum spanning tree** (MST)
  - Spanning tree *T* that connects all vertices minimizing total cost $w(T) = \sum_{(u,v) \in T} w(u,v)$
- How to find a MST?
  - Optimization problem



Example 1

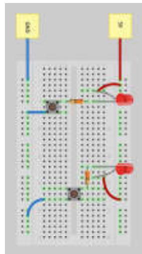
- **Road Problem**
  - A town has a set of houses and a set of roads
  - A road connects 2 and only 2 houses
  - A road connecting houses u and v has a repair cost w(u, v)



- **Goal:** Repair enough roads such that:
  - Everyone stays connected
    - Can reach every house from all other houses
  - Total repair cost is minimum

Example 2

- **Electronic circuit problem**
  - Interconnect the pins of n components in an electronic circuit
  - It is possible to arrange n - 1 wires, each connecting two pins
  - **Goal**: Use the least amount of wire
    - The wiring problem can be modeled by an undirected graph G = (V,E)
      - V: set of pins
      - E: set of possible interconnections between pairs of pins
      - For each edge (u,v), there is a cost (amount of wire) to connect u and v



André de Carvalho - ICMC/USP 7

Minimum Spanning Trees

- Other applications
  - Clustering
    - K-means: find MST and remove the k-1 most expensive edges
  - Design of Network
    - Cable TV, distributed systems, electrical, hydraulic, Road, Streets
  - Taxonomy
    - Animals, genes,
  - Travelling salesman problem

André de Carvalho - ICMC/USP 8

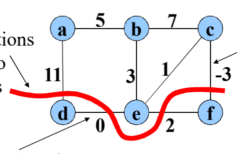
Cutting a graph

No edge in the edge set A **crosses** the cut

The cut **respects** A = {(a, b), (b, c)}

Light edge: minimum weight edge **crossing** the cut

A **cut** partitions vertices into disjoint sets *S* and *V* – *S*



A **light edge crossing** the cut (can be more than one)

A edge **crossing** the cut

One endpoint is in *S* and the other is in *V* – *S*

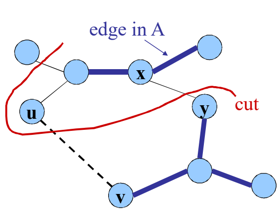
André de Carvalho - ICMC/USP 9

Safe edge recognition rule

- Theorem 23.1:
  - Let (S, V-S) be any cut that respects A and let (u, v) be a light edge crossing (S, V-S)
  - Then (u, v) is safe for A
- Proof:
  - Let T be a MST that includes A
  - **Case 1:** (u, v) in T
    - It is proved
  - **Case 2:** (u, v) not in T
    - See next

André de Carvalho - ICMC/USP 10

Proof of case 2



Shows edges in T

Edge (x, y) crosses the cut

Let T´ = T - {(x, y)} ∪ {(u, v)}

Since (u, v) is an light edge for cut, w(u, v) ≤ w(x, y)

Thus w(T´) = w(T) - w(x, y) + w(u, v) ⇒ w(T´) ≤ w(T)

But T is a MST, thus w(T) ≤ w(T´)

Hence, T´ is also a MST

Thus, (u, v) is safe for A

André de Carvalho - ICMC/USP 11

Greedy Algorithm for MST

- Greedy algorithm:
  - Other algorithm design technique
    - Anther is dynamic programming
  - At each step, select, from the possible options, the best option at the moment
  - Not guaranteed to find globally optimal solutions
- For the MST problem, some greedy algorithms can find globally optimal solutions
  - Algorithm grows the MST one edge at the time
  - Manages a set of edges A with the loop invariant:
    - Before each iteration, A is s subset of some MST

André de Carvalho - ICMC/USP 12

Generic MST Algorithm

```
Generic-MST(G, w)
1 A←∅  // Contains edges that belong to a MST
2 while A does not form a spanning tree do
3    Find an edge (u,v) that is safe for A
4    A ← A∪{(u,v)}
5 return A
```

***Safe edge*:** can be added to A maintaining the invariant

```
MoreSpecific-MST(G, w)
1    A←∅  // Contains edges that belong to a MST
2    while A does not form a spanning tree do
3.1    Make a cut (S, V-S) of G that respects A
3.2    Take the min-weight edge (u,v) connecting S to V-S
4    A ← A∪{(u,v)}
5 return A
```

André de Carvalho - ICMC/USP 13

Generic MST Algorithm

- Initialization: after line 1, set A satisfies the loop
- Maintenance: loop in lines 2-4 keep the invariant by adding only safe edges
- Termination: all edges added to A are in a MST
  - Therefore, A returned in line 5 is a MST
- Challenge: find a safe edge for line 3

André de Carvalho - ICMC/USP 14

Disjoint sets

- Groups *n* distinct elements into a collection of disjoint sets S
  - Each disjoint set is identified by one of its members, called a representative
    - In a forest of trees, each set can be a tree and each element a vertex in a tree
- Two important operations are:
  - Find to which set a given element belongs
  - Unite two sets creating a new set

André de Carvalho - ICMC/USP 15

Operations for disjoint sets

- MAKE-SET*(x)* creates a new set whose only member (and thus representative) is *x*
  - Since the sets are disjoint, *x must* not already be in some other set from S (collection of sets)
- UNION*(x, y)* unites the sets that contain *x* and *y* into a new set, the union of these two sets
  - The two sets, S_x and S_y, are assumed to be disjoint
    - The representative of the united set is any member of $S_x \cup S_y$
    - The two previous sets are removed the collection S
- FIND-SET*(x)* returns a representative of the (unique) set containing *x*

André de Carvalho - ICMC/USP 16

MST Algorithms

- Two MST algorithms are often employed
  - Use different rules to find a safe edge in line 3
  - Kruskal
    - The set of edges A forms a forest of trees
    - The safe edged added to A is always the least-weight edge in the graph connecting two components
  - Prim-Jarnik (Prim)
    - The set of edges A forms a single tree
    - The safe edged added to A is always the least-weight edge in the graph connecting the tree to a vertex outside the tree

André de Carvalho - ICMC/USP 17

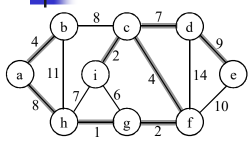
Kruskal Algorithm

- Keeps adding the edge with the smallest weight that connects two trees of the forest
  - One at a time

```
MST-Kruskal (G, w, r)
1 A ← ∅
2 for each v ∈ V[G] do
3    MAKE-SET (v)
4 sort the edges of E into nondecreasing order by weight
5 for each edge (u,v) ∈ E taken in nondecreasing by weight do
6    if FIND-SET(u) ≠ FIND-SET (v)
7    then A ← A ∪ {(u,v)}
8          UNION (u,v)
9 return A
```

André de Carvalho - ICMC/USP 18

Exemple



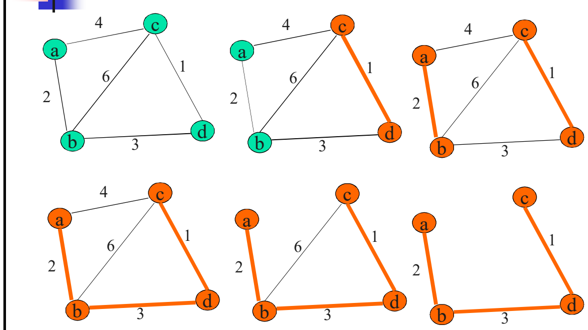
MAKE-SET
{a}, {b}, {c}, {d}, {e}, {f}, {g}, {h}, {i}

Sort Edges

1: (h, g)
2: (c, i), (g, f)
4: (a, b), (c, f)
6: (i, g)
7: (c, d), (i, h)
8: (a, h), (b, c)
9: (d, e)
10: (e, f)
11: (b, h)
14: (d, f)

	$A \leftarrow A \cup \{(u,v)\}$	*UNION (u,v)*
1.	Add (h, g)	{g, h}, {a}, {b}, {c}, {d}, {e}, {f}, {i}
2.	Add (c, i)	{g, h}, {c, i}, {a}, {b}, {d}, {e}, {f}
3.	Add (g, f)	{g, h, f}, {c, i}, {a}, {b}, {d}, {e}
4.	Add (a, b)	{g, h, f}, {c, i}, {a, b}, {d}, {e}
5.	Add (c, f)	{g, h, f, c, i}, {a, b}, {d}, {e}
6.	Ignore (i, g)	{g, h, f, c, i}, {a, b}, {d}, {e}
7.	Add (c, d)	{g, h, f, c, i, d}, {a, b}, {e}
8.	Ignore (i, h)	{g, h, f, c, i, d}, {a, b}, {e}
9.	Add (a, h)	{g, h, f, c, i, d, a, b}, {e}
10.	Ignore (b, c)	{g, h, f, c, i, d, a, b}, {e}
11.	Add (d, e)	{g, h, f, c, i, d, a, b, e}
12.	Ignore (e, f)	{g, h, f, c, i, d, a, b, e}
13.	Ignore (b, h)	{g, h, f, c, i, d, a, b, e}
14.	Ignore (d, f)	{g, h, f, c, i, d, a, b, e}

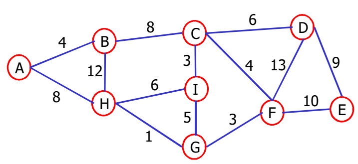
Example 2



André de Carvalho - ICMC/USP 20

Exercise

- Apply Kruskak to the graph below (r = B)



- Keep track of:
  - What is the set *A*, what is a collection *S*, what cuts did you make

André de Carvalho - ICMC/USP 21

Kruskal Running Time

- Initialization $O(V)$ time
- Sorting the edges $\Theta(E \lg E) = \Theta(E \lg V)$ (why?)
- $O(E)$ calls to FindSet
- Union operation costs
  - Let $t(v)$ be the number of times v is moved to a new set (cluster)
  - Each time a vertex is moved to a new set, the size of the new set at least doubles: $t(v) \leq \log V$
  - Total time spent doing Union: $\sum_{v \in V} t(v) \leq |V| \log |V|$
- Total time: $O(E \lg V)$

André de Carvalho - ICMC/USP 22

Prim-Jarnik Algorithm

- Vertex based algorithm
- Grows one tree T, one vertex at a time
- Stores the portion of T already computed in a set A
- Labels the vertices *v* outside of the set *A* with **key**[v]
  - The minimum weigth of an edge connecting *v* to a vertex in *A*
  - **key**[v] = ∞, if no such edge exists

André de Carvalho - ICMC/USP 23

Priority Queue

- Data structure to maintain a set *S* of elements, each with an associated value called a ***key***
  - A ***min-priority queue*** supports operations:
    - INSERT*(S, x)* inserts the element *x* into the set *S*. This operation could be written as $S \leftarrow S \cup \{x\}$
    - MINIMUM*(S)* returns the element of *S* with the smallest key
    - EXTRACT-MIN*(S)* removes and returns the element of *S* with the smallest key
    - DECREASE-KEY*(S, x, k)* decreases the value of element *x*' s key to the new value *k*

André de Carvalho - ICMC/USP 24

Prim-Jarnik Algorithm

Complexity:
Using binary heaps: O(E lg V)
- Initialization: O(V)
- Building initial queue: O(V)
- V Extract-Mins: O(V lgV)
- E Decrease-Keys: O(E lg V)

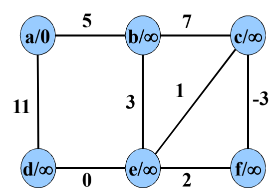
Using Fibonacci heaps: O(E + V lg V)

```
MST-Prim (G, w, r)
for each u ∈ V[G] do
    key[u] ← ∞
    π[u] ← NIL
key[r] ← 0
Q ← V[G]   // Q is a priority queue ADT
while Q ≠ ∅ do
  u ← EXTRACT-MIN(Q)  // make u part of T
  for each v ∈ Adj[u] do
     if v ∈ Q and w(u, v) < key[v]   // update keys
     then π[v] ← u;
          key[v] ← w(u, v) // decrease key
```

Note: A = {(v, π[v]) : v ∈ v - {r} - Q}.

André de Carvalho - ICMC/USP 25

Example of Prim Algorithm

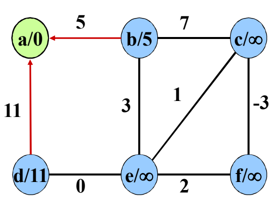


Not in tree T

Q = a b c d e f
0 ∞ ←——→ ∞

André de Carvalho - ICMC/USP 26

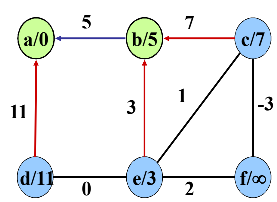
Example of Prim Algorithm



Q = b d c e f
5 11 ∞←→∞

André de Carvalho - ICMC/USP 27

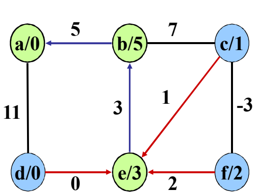
Example of Prim Algorithm



Q = e c d f
3 7 11 ∞

André de Carvalho - ICMC/USP 28

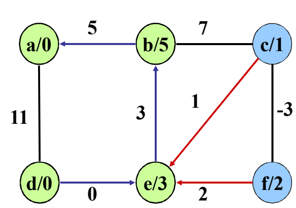
Example of Prim Algorithm



Q = d c f
0 1 2

André de Carvalho - ICMC/USP 29

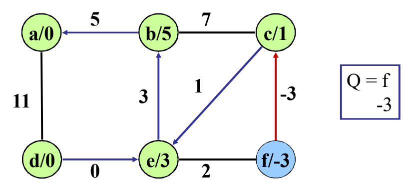
Example of Prim Algorithm



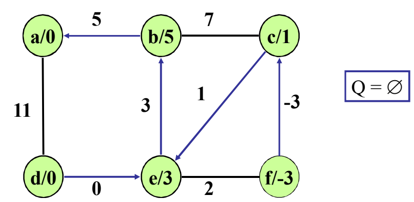
Q = c f
1 2

André de Carvalho - ICMC/USP 30

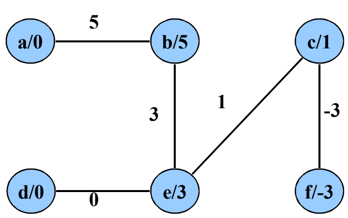
Example of Prim Algorithm



Example of Prim Algorithm

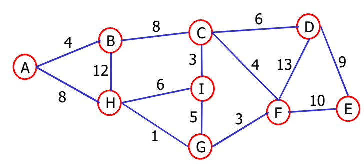


Example of Prim Algorithm



Prim-Jarnik's Example

- Apply Prim to the graph below (r = B)



- Keep track of:
  - What is set *A*, which vertices are in *Q*, what cuts are we making, what are the *key* values

Prim-Jarnik's Running Time

- Time = $|V| T(\text{extractMin}) + O(E) T(\text{modifyKey})$
- Binary heap implementation:
  - Time = $O(V \lg V + E \lg V) = O(E \lg V)$

Q	T(extractMin)	T(modifyKey)	Total
array	$O(V)$	$O(1)$	$O(V^2)$
binary heap	$O(\lg V)$	$O(\lg V)$	$O(E \lg V)$
Fibonacci heap	$O(\lg V)$	$O(1)$ amortized	$O(V \lg V + E)$

Next Lecture

- Single-source shortest paths in weighted graphs
  - Shortest-Path Problems
  - Dijkstra's Algorithm
  - Bellman-Ford Algorithm
  - Shortest-Paths in DAG's

Acknowledgement

- A large part of this material were adapted from
  - Simonas Šaltenis, Algorithms and Data Structures, Aalborg University, Denmark
  - Mary Wootters, Design and Analysis of Algorithms, Stanford University, USA
  - George Bebis, Analysis of Algorithms CS 477/677, University of Nevada, Reno
  - David A. Plaisted, Information Comp 550-001, University of North Carolina at Chapel Hill

André de Carvalho - ICMC/USP 37

Questions

André de Carvalho - ICMC/USP 38