Algorithm Design and Analysis

วิชาบังคับก่อน: 204251 และ 206281
ผู้สอน: ตอน 1 ผศ. เบญจมาศ ปัญญางาม เรียน ห้อง 207
ตอน 2 ผศ. ดร. จักริน ชวชาติ เรียน ห้อง 209

บทที่ 8
อัลกอริทึมกรีดดี
(Greedy algorithms)
Part I

Greedy algorithms

Contents

- การจับคู่ที่มั่นคง (Stable marriages)
- อัลกอริทึมครูสคัลและการพิสูจน์ความถูกต้อง (Kruskal's algorithm and its correctness)
- อัลกอริทึมไดสตรา (Dijkstra's algorithm)
- การจัดสรรทรัพยากร (Resource scheduling)

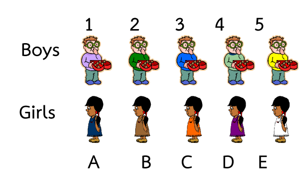
Greedy Algorithms

- The greedy-choice property: a globally optimal solution can be arrived at by making a locally optimal (greedy) choice.
  - Always makes the choice that looks best at the moment
  - Sometimes, works well for optimization problems

Stable Marriage Problem

- There are n men and n women, which are unmarried.
  - Each has a preference list on the persons of the opposite sex
  - Does there exist and can we find a stable matching ?

Stable matching : a matching of men and women, such that there is no pair of a man and a woman who both prefer each other above their partner in the matching?



Preference lists

Boys	Girls
1: CBEAD	A : 35214
2 : ABECD	B : 52143
3 : DCBAE	C : 43512
4 : ACDBE	D : 12345
5 : ABDEC	E : 23415

Stable Marriage

Try "greedy" strategy for boys

First :

Marry Boy 1 with Girl C

(his 1st choice)

1: CBEAD
2 : ABECD
3 : DCBAE
4 : ACDBE
5 : ABDEC

1

C

Preference lists

Stable Marriage

Try "greedy" strategy for boys

First :

Marry Boy 1 with Girl C

(his 1st choice)

1 C
2 : ABED
3 : DBAE
4 : ADBE
5 : ABDE

1

C

Preference lists

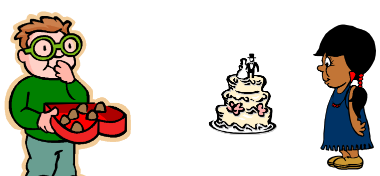
Stable Marriage

1 C
2 A
3 : DBE
4 : DBE
5 : BDE

Second :

Marry Boy 2 with Girl A:

(best remaining choice)



2

A

Preference lists

Stable Marriage

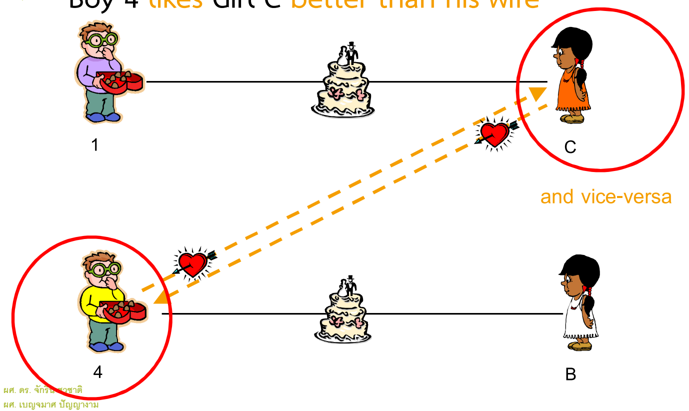
1 C
2 A
3 D
4 B
5 E

Final "boy greedy" marriages

Preferences

Stable Marriage

- Trouble!
  - Boy 4 likes Girl C better than his wife



Stable Marriage

- Gale-Shapley algorithm uses greedy approach that
  - finds always a stable matching
- Idea :
  - Fix some ordering on the boys
  - Repeat until everyone is matched
    - Let X be the first unmatched boy in the ordering
    - Find girl Y such that Y is the most desirable girl in X's list
      - case 1, Y is unmatched, or
      - case 2 , Y is currently matched to a Z and X is more preferable to Y than Z (so, change Z to be unmatched)
    - Match X and Y

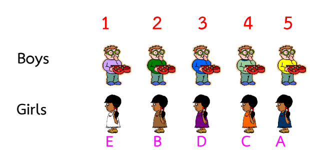
Stable Marriage

- Gale-Shapley algorithm uses greedy approach that
  - finds always a stable matching

Preference lists

Boys	Girls
1: CBEAD	A : 35214
2 : ABECD	B : 52143
3 : DCBAE	C : 43512
4 : ACDBE	D : 12345
5 : ABDEC	E : 23415

Matching result



Greedy algorithms in Graph Problem

- Finding a minimum-cost spanning tree
  - Kruskal's algorithm
    - Always tries the *lowest-cost* remaining edge
  - Prim's algorithm
    - Always takes the *lowest-cost* edge between nodes in the spanning tree and nodes not yet in the spanning tree
- Finding the shortest path in a graph : Dijkstra's algorithm.
  - Always takes the *shortest* edge connecting a known node to an unknown node

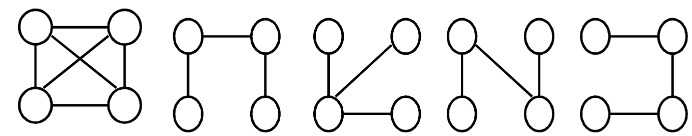
Minimum-Cost Spanning Tree Problem

A minimum spanning tree (MST) is a least-cost subset of the edges of a graph that connects all the nodes

- Suppose you want to supply a set of houses (say, in a new subdivision) with electric power, water or telephone lines
  - To reduce costs even further, you could connect the houses with a *minimum-cost* spanning tree
- Applications of Minimal Spanning Trees
  - Network design examples
    - Telephone, electrical, TV cable, Computer, road
  - ....

Graph & Spanning Tree

Spanning Trees: a subgraph T of an undirected graph G = (V;E) is a spanning tree of G if it is a tree and contains every vertex of G



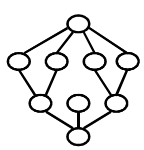
A connected, undirected graph

The spanning tree of the graph

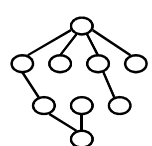
➔ a connected subgraph in which there are no cycles

Finding a spanning tree

- To find a spanning tree of a graph,
  - Pick an initial node
  - Do a search from the initial node:
  - find a node that is not in the spanning tree,
  - add to the spanning tree both the new node *and* the edge

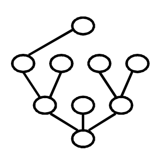


An undirected graph



Starting from top

BFS



DFS

Minimum Spanning Tree Problem

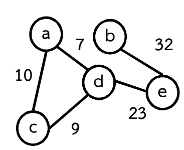
Given: a connected weighted undirected graph,

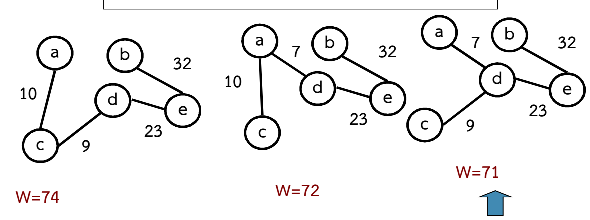
Find: a minimum(weighted) spanning tree, *T, of G*

Example:

Three forms of spanning trees

G





A minimum-cost (weight) spanning tree (MST)

Weighted Graphs: A weighted graph is a graph, in which each edge has a weight (some real number).

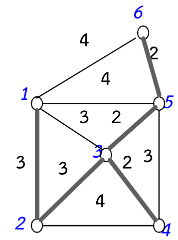
Weight of a Graph: The sum of the weights of all edges.

Minimum Spanning Tree Problem : Solution

- Basically, the choice (u,v) is a safe choice to add so that A can still be extended to form an MST.

Note that if A is feasible it cannot contain a cycle.

- A generic greedy algorithm operates by repeatedly adding any *safe edge* to the current spanning tree.
  - Stop when all nodes have been added to the tree
- Two greedy algorithms (differ in how to choose edges to add) for finding a MST:
  - Kruskal's Algorithm
  - Prim's Algorithm



The result is a least-cost (3+3+2+2+2=12)

MST

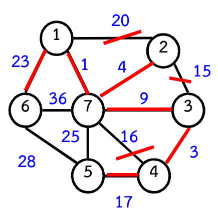
MST Problem : Solution by Kruskal's algorithm

- Idea is to start with an empty graph and try to add edges one at a time,always making sure that what is built remains acyclic.
- Finding an edge of lowest cost can be done just by sorting the edges or built min heap

```
MST-Kruskal(G)
T = ∅        // starting with empty spanning tree
Construct a priority queue Q containing all edges in E //|E| log |E|
VS = ∅ ,  For each vertex v  ∈ V  do  add {v} to VS    //O(V)
While | VS | > 1   //O(E log E)
    Choose (v,w) an edge in Q of lowest cost
    delete (v,w) from Q     //Q=Q -{(v,w)}, use log |E| (if min-heap Q)
    if v and w are in different set w1 and w2 in VS then
        replace w1 and w2 in VS by w1 ∪ w2
        add (v,w) to T              //T = T ∪ {(v,w)}
// T is now a MST
```

Total running time is $O(E \log E)$.

MST Problem : Solution by Kruskal's algorithm



Edge	action	VS
$\{v_1,v_7\}$	add	$\{v_1,v_7\}\{v_2\}\{v_3\}\{v_4\}\{v_5\}\{v_6\}$
$\{v_3,v_4\}$	add	$\{v_1,v_7\}\{v_2\}\{v_3,v_4\}\{v_5\}\{v_6\}$
$\{v_2,v_7\}$	add	$\{v_1,v_2,v_7\}\{v_3,v_4\}\{v_5\}\{v_6\}$
$\{v_3,v_7\}$	add	$\{v_1,v_2,v_3,v_4,v_7\}\{v_5\}\{v_6\}$
$\{v_2,v_3\}$	reject	
$\{v_4,v_7\}$	reject	
$\{v_4,v_5\}$	add	$\{v_1,v_2,v_3,v_4,v_5,v_7\}\{v_6\}$
$\{v_1,v_2\}$	reject	
$\{v_1,v_6\}$	add	$\{v_1,v_2,v_3,v_4,v_7,v_5,v_6,v_7\}$

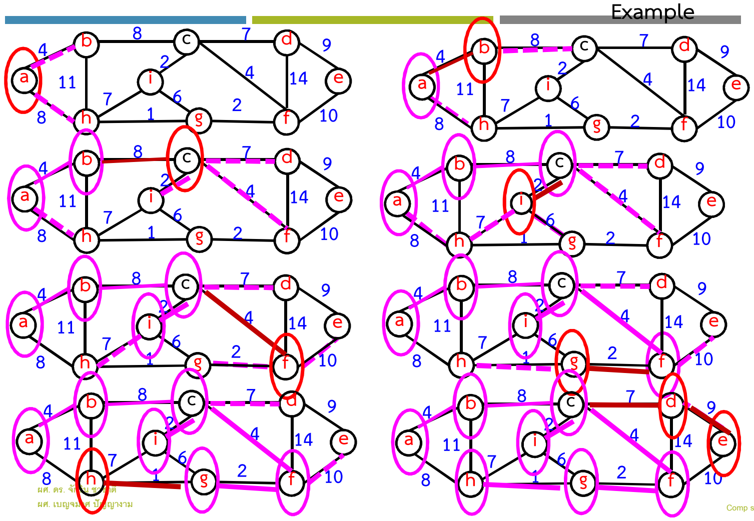
MST Problem : Solution by Prim's algorithm

- Start with any *one node* in the spanning tree, and repeatedly add the lowest cost edge, and the node it leads to, for which the node is not already in the spanning tree

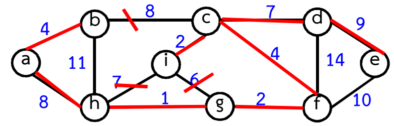
```
T = a spanning tree containing a single node s;
E = set of edges adjacent to s;
while T does not contain all the nodes {
    remove an edge (v, w) of lowest cost from E
    if w is already in T   then  discard edge (v, w)
    else
        add edge (v, w) and node w to T
          add to E the edges adjacent to w
}
```

- An edge of lowest cost can be found with a priority queue

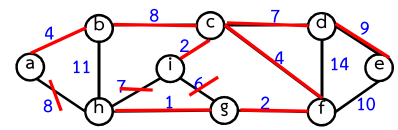
MST Problem : Solution by Prim’s algorithm: Example



MST Problem : Solutions

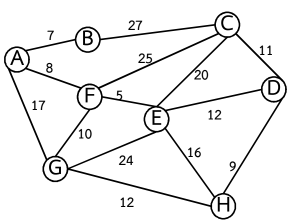


- The minimum spanning tree may not be unique. However, if the weights of all the edges are pairwise distinct, it is indeed unique.



MST Problem :Practice & Solution

- Solution by Kruskal’s Algorithm



MST Problem : Practice & Solution

- Solution by Prim’s Algorithm