Design and Analysis of Algorithms
Approximation algorithms for NP-complete problems

Haidong Xue

Summer 2012, at GSU

Approximation algorithms for NPC problems

- If a problem is NP-complete, there is very likely no polynomial-time algorithm to find an optimal solution
- The idea of approximation algorithms is to develop polynomial-time algorithms to find a near optimal solution

Approximation algorithms for NPC problems

- E.g.: develop a greedy algorithm without proving the greedy choice property and optimal substructure.
- Are those solution found near-optimal?
- How near are they?

Approximation algorithms for NPC problems

- **Approximation ratio $\boldsymbol{\rho(n)}$**
  – Define the cost of the optimal solution as C*
  – The cost of the solution produced by a approximation algorithm is C
  – $\boldsymbol{\rho(n)} \geq max(\frac{C}{C^*}, \frac{C^*}{C})$
- The approximation algorithm is then called a $\boldsymbol{\rho(n)}$-approximation algorithm.

Approximation algorithms for NPC problems

- E.g.:
  - If the total weigh of a MST of graph G is 20
  - A algorithm can produce some spanning trees, and they are not MSTs, but their total weights are always smaller than 25
  - What is the approximation ratio?
    - 25/20 = 1.25
  - This algorithm is called?
    - A 1.25-approximation algorithm

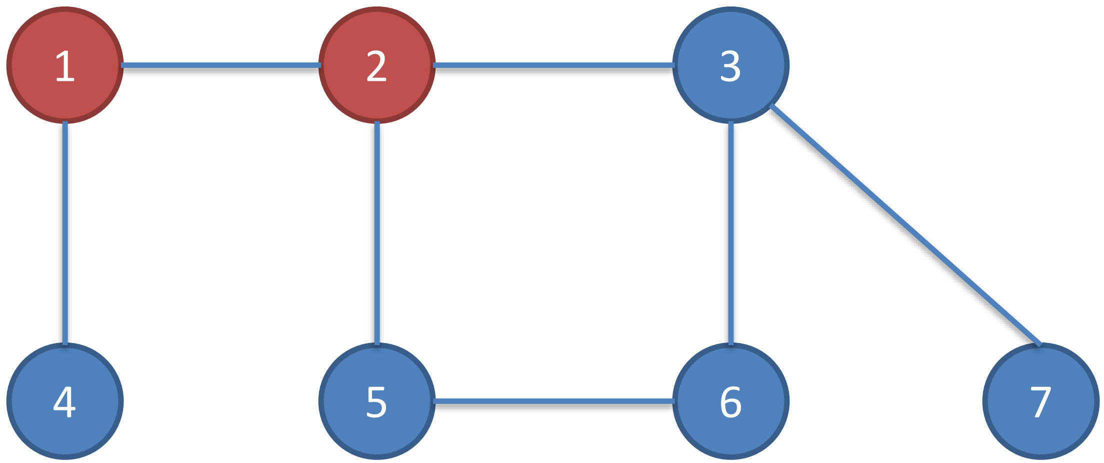
Approximation algorithms for NPC problems

- What if $\boldsymbol{\rho(n)}$=1?
- It is an algorithm that can always find a optimal solution

Vertex-cover problem and a 2-approximation algorithm

- What is a vertex-cover?
- Given a undirected graph G=(V, E), **vertex-cover** V':
  - $V' \subseteq V$
  - for each edge (u, v) in E, either $u \in V'$ or $v \in V'$ (or both)
- The size of a vertex-cover is |V'|

Vertex-cover problem and a 2-approximation algorithm

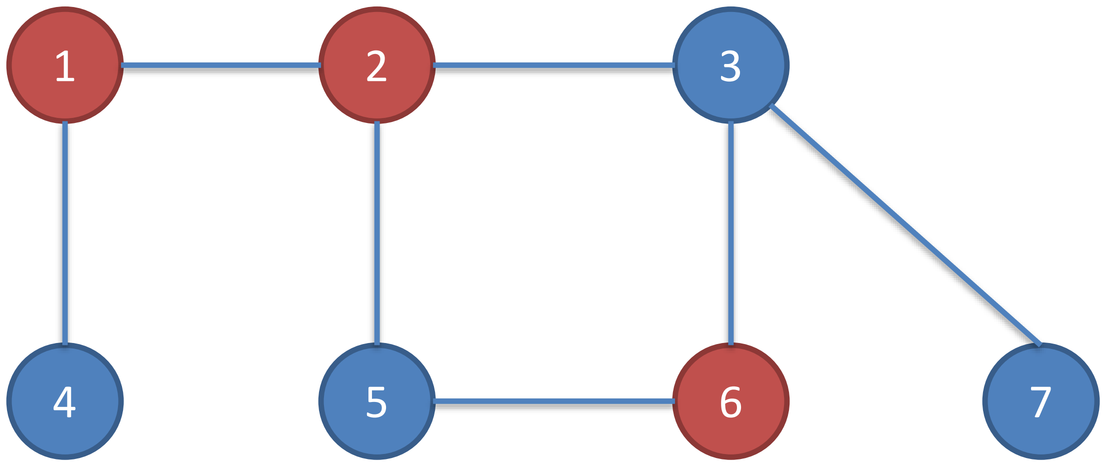


Are the red vertices a vertex-cover?

No. why?

Edges (5, 6), (3, 6) and (3, 7) are not covered by it

Vertex-cover problem and a 2-approximation algorithm

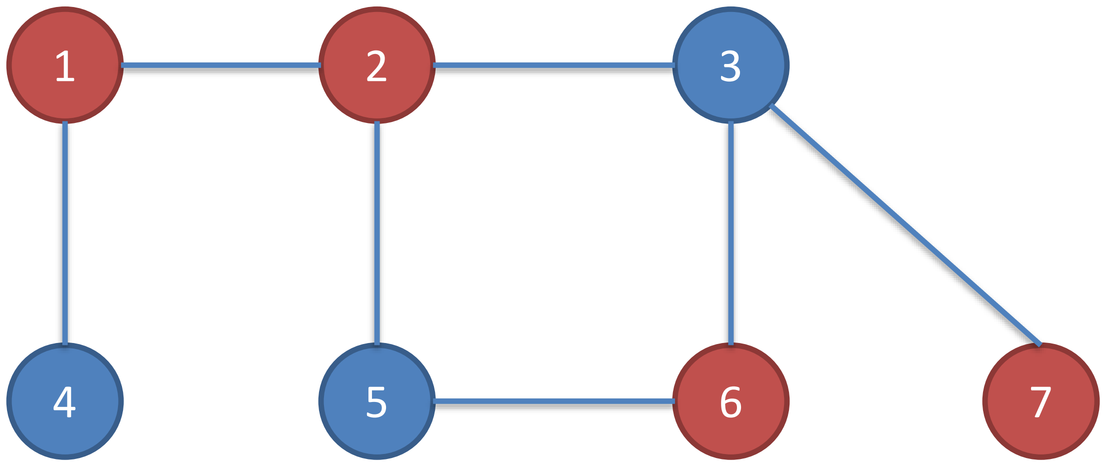


Are the red vertices a vertex-cover?

No. why?

Edge (3, 7) is not covered by it

Vertex-cover problem and a 2-approximation algorithm



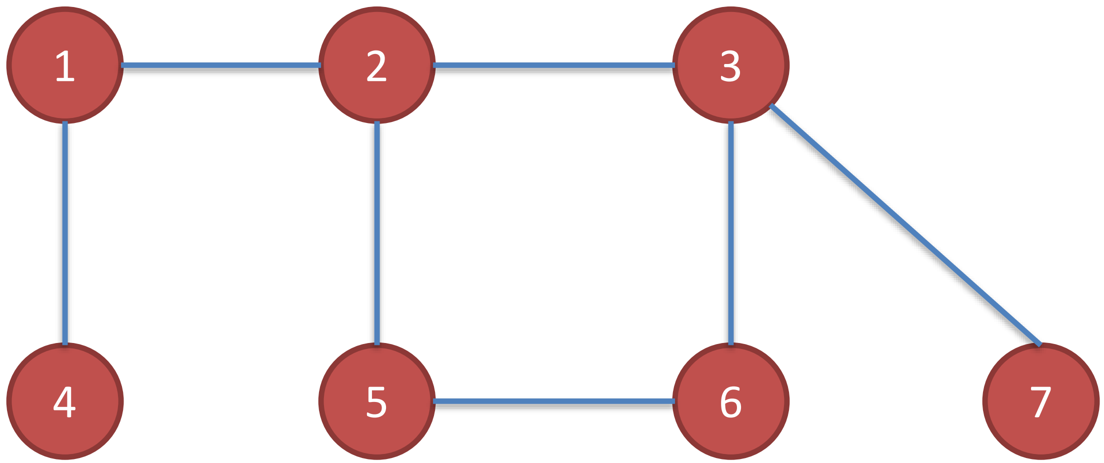
Are the red vertices a vertex-cover?

Yes

What is the size?

4

Vertex-cover problem and a 2-approximation algorithm



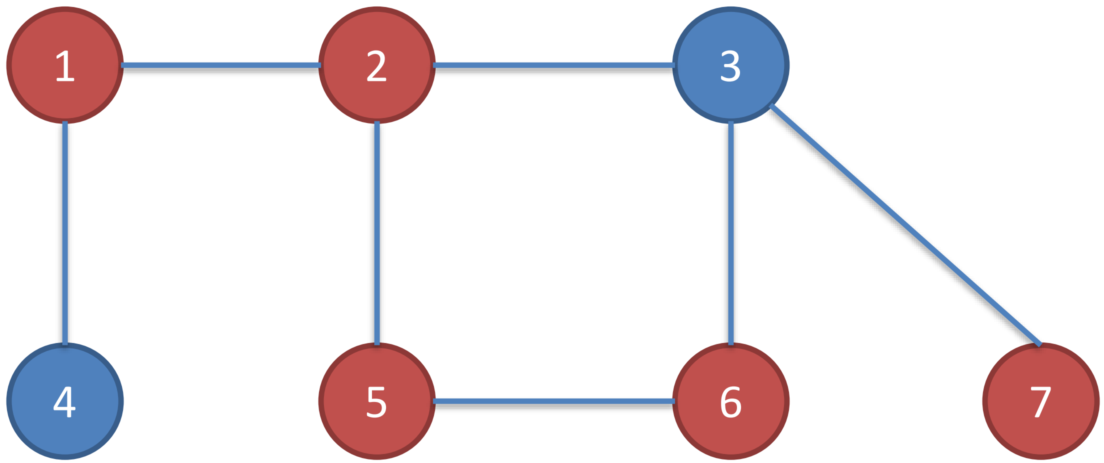
Are the red vertices a vertex-cover?

Yes

What is the size?

7

Vertex-cover problem
and a 2-approximation algorithm



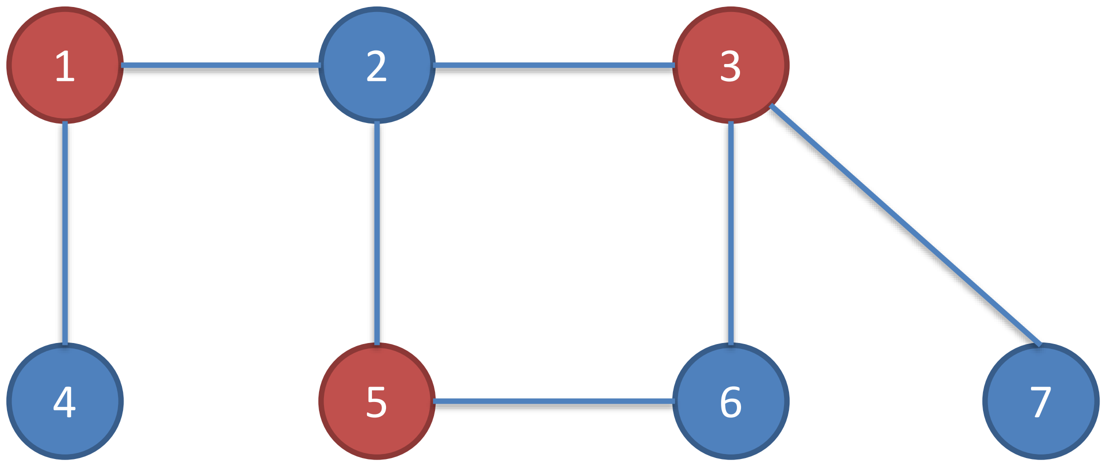
Are the red vertices a vertex-cover?

Yes

What is the size?

5

Vertex-cover problem and a 2-approximation algorithm



Are the red vertices a vertex-cover?

Yes

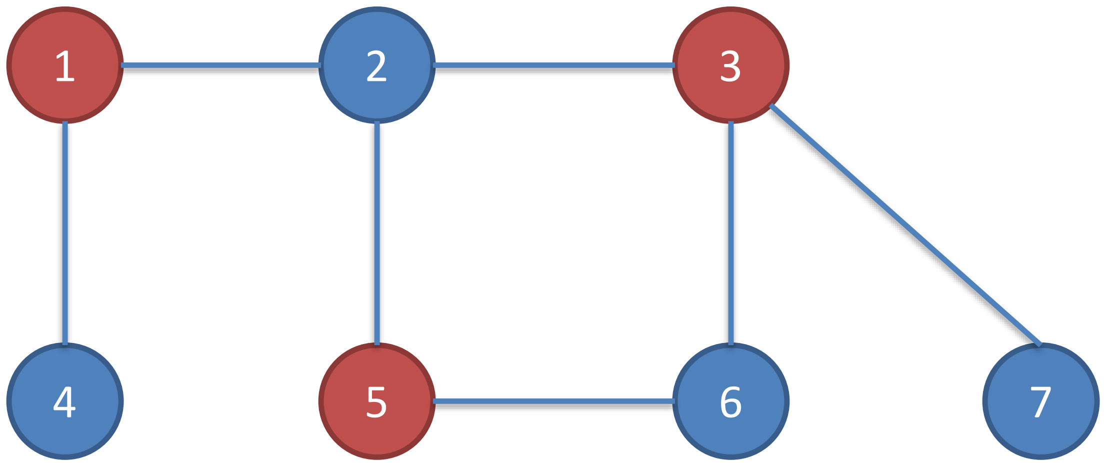
What is the size?

3

Vertex-cover problem and a 2-approximation algorithm

- **Vertex-cover problem**
  - Given a undirected graph, find a vertex cover with minimum size.

Vertex-cover problem and a 2-approximation algorithm



A minimum vertex-cover

Vertex-cover problem and a 2-approximation algorithm

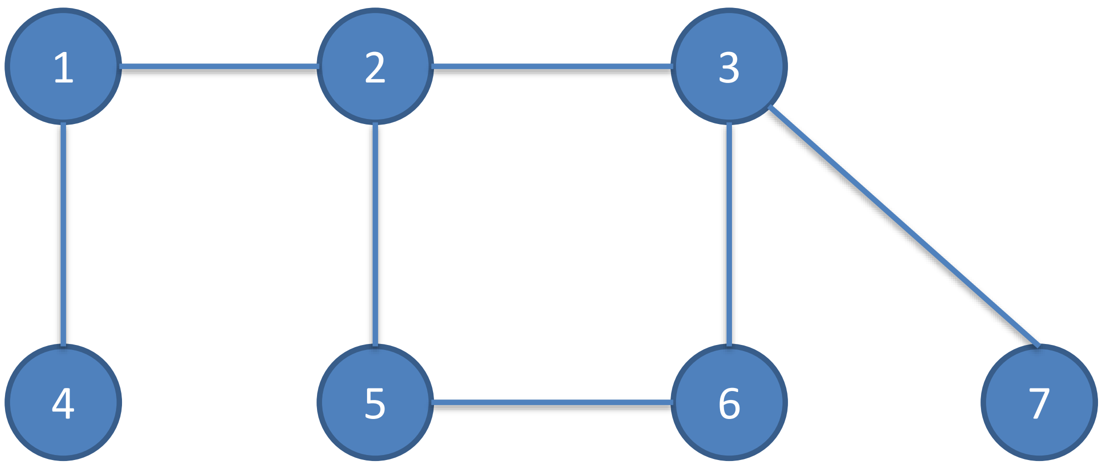
- Vertex-cover problem is **NP-complete**
- A 2-approximation polynomial time algorithm is as the following:
- **APPROX-VERTEX-COVER**(G)

```
C = ∅;
E'=G.E;
while(E' ≠ ∅ ){
    Randomly choose a edge (u,v) in E', put u and v into C;
    Remove all the edges that covered by u or v from E'
}
Return C;
```

Vertex-cover problem and a 2-approximation algorithm

APPROX-VERTEX-COVER(G)

    C = ∅;
    E'=G.E;
    while(E' ≠ ∅){
        Randomly choose a edge (u,v) in E', put u and v into C;
        Remove all the edges that covered by u or v from E'
    }
    Return C;



Vertex-cover problem and a 2-approximation algorithm

APPROX-VERTEX-COVER(G)

$C = \emptyset$;

E'=G.E;

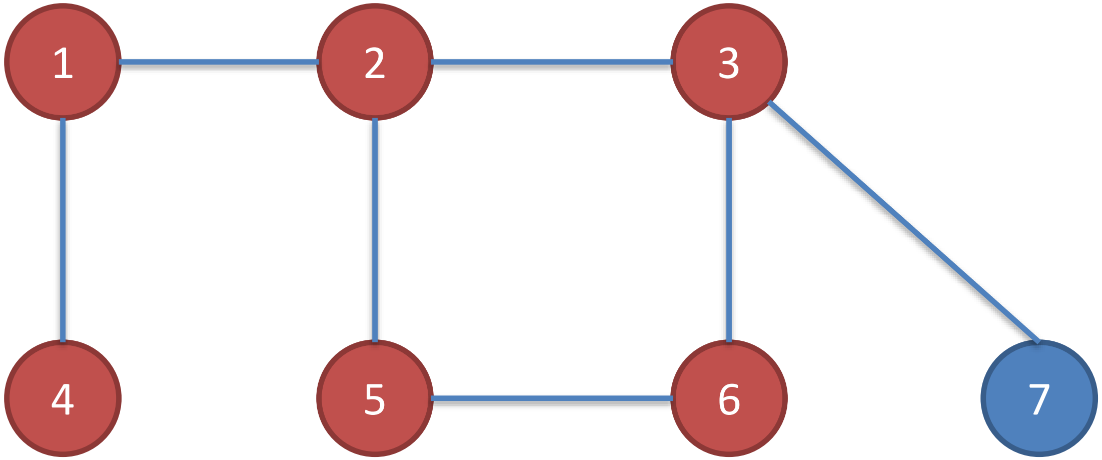
while(E' $\neq \emptyset$){

Randomly choose a edge (u,v) in E', put u and v into C;

Remove all the edges that covered by u or v from E'

}

Return C;



It is then a vertex cover

Size? 6

How far from optimal one? Max(6/3, 3/6) = 2

Vertex-cover problem and a 2-approximation algorithm

APPROX-VERTEX-COVER(G)

C $= \emptyset$;

E’=G.E;

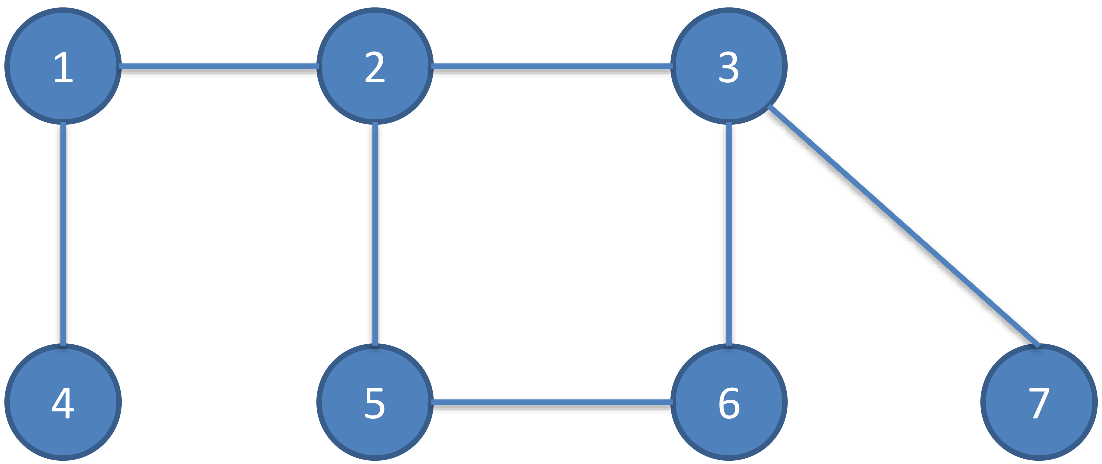
while(E’ $\neq \emptyset$){

Randomly choose a edge (u,v) in E’, put u and v into C;

Remove all the edges that covered by u or v from E’

}

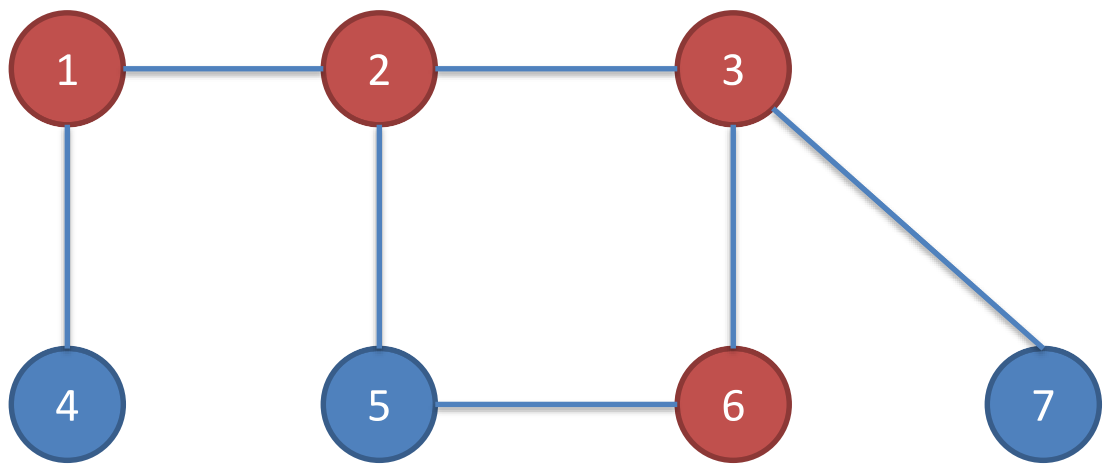
Return C;



Vertex-cover problem and a 2-approximation algorithm

APPROX-VERTEX-COVER(G)

$C = \emptyset$;
E'=G.E;
while(E' $\neq \emptyset$){
Randomly choose a edge (u,v) in E', put u and v into C;
Remove all the edges that covered by u or v from E'
}
Return C;



It is then a vertex cover

Size? 4

How far from optimal one? Max(4/3, 3/4) = 1.33

Vertex-cover problem and a 2-approximation algorithm

- **APPROX-VERTEX-COVER**(G) is a 2-approximation algorithm
- When the size of minimum vertex-cover is *s*
- The vertex-cover produced by **APPROX-VERTEX-COVER** is at most 2*s*

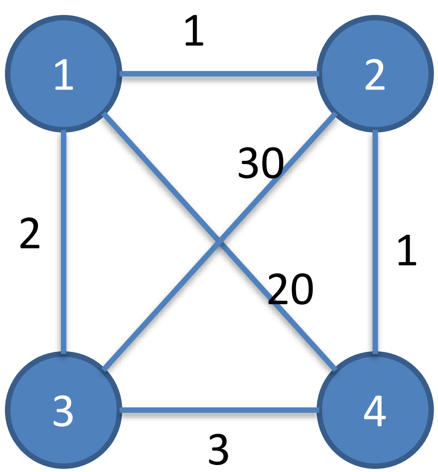
Vertex-cover problem and a 2-approximation algorithm

Proof:

- Assume a minimum vertex-cover is U*
- A vertex-cover produced by **APPROX-VERTEX-COVER**(G) is U
- The edges chosen in **APPROX-VERTEX-COVER**(G) is A
- A vertex in U* can only cover 1 edge in A
  - So |U*|>= |A|
- For each edge in A, there are 2 vertices in U
  - So |U|=2|A|
- So |U*|>= |U|/2
- So $\frac{|\mathrm{U}|}{|\mathrm{U}^*|} \leq 2$

Traveling-salesman problem

- **Traveling-salesman problem (TSP):**
  - Given a weighted, undirected graph, start from certain vertex, find a **minimum** route visit each vertices once, and return to the original vertex.



Traveling-salesman problem

- TSP is a NP-complete problem
- There is **no polynomial-time approximation** algorithm with a **constant approximation ratio**
- Another strategy to solve NPC problem:
  - **Solve a special case**

Traveling-salesman problem

- **Triangle inequality**:
  - Weight(u, v) <= Weight(u, w) + Weight(w, v)
- E.g.:
  - If all the edges are defined as the distance on a 2D map, the triangle inequality is true
- For the TSPs where the triangle inequality is true:
  - There is a 2-approximation polynomial time algorithm

Traveling-salesman problem

APPROX-TSP-TOUR(G)

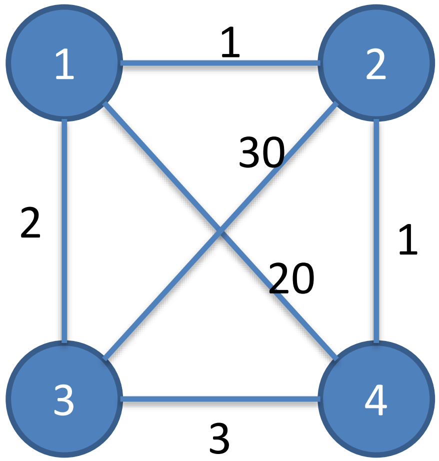
Find a MST m;

Choose a vertex as root r;

return preorderTreeWalk(m, r);

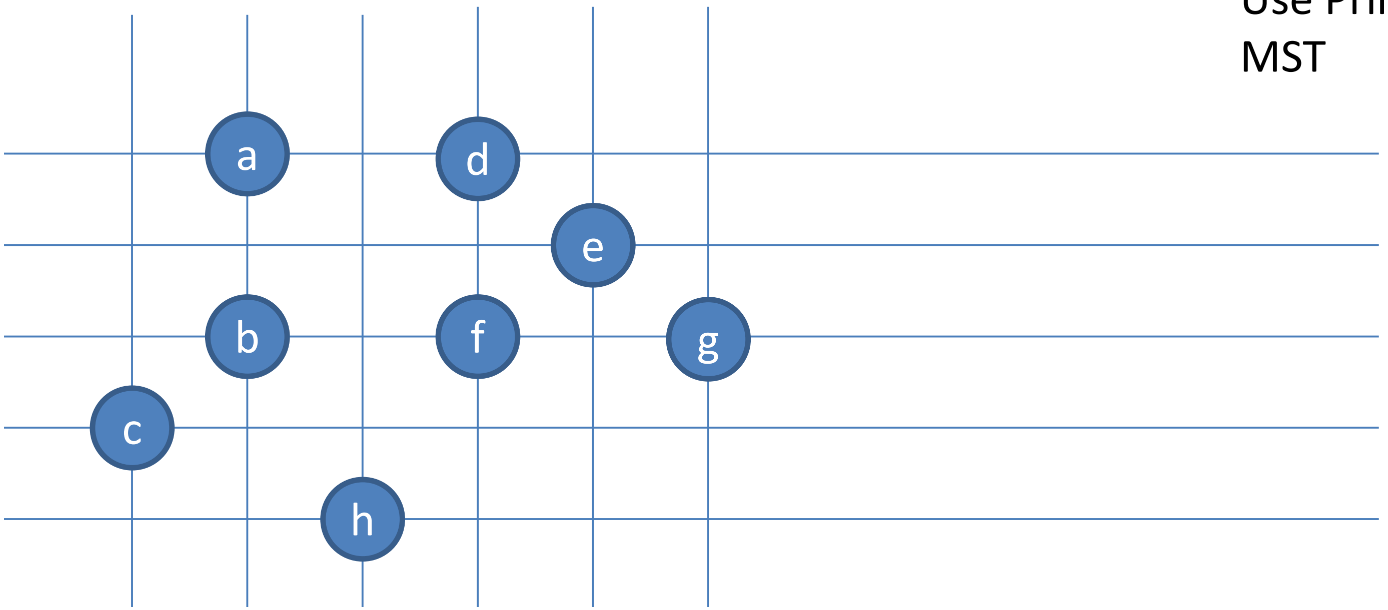
Traveling-salesman problem

Can we apply the approximation algorithm on this one?



No. The triangle inequality is violated.

Traveling-salesman problem

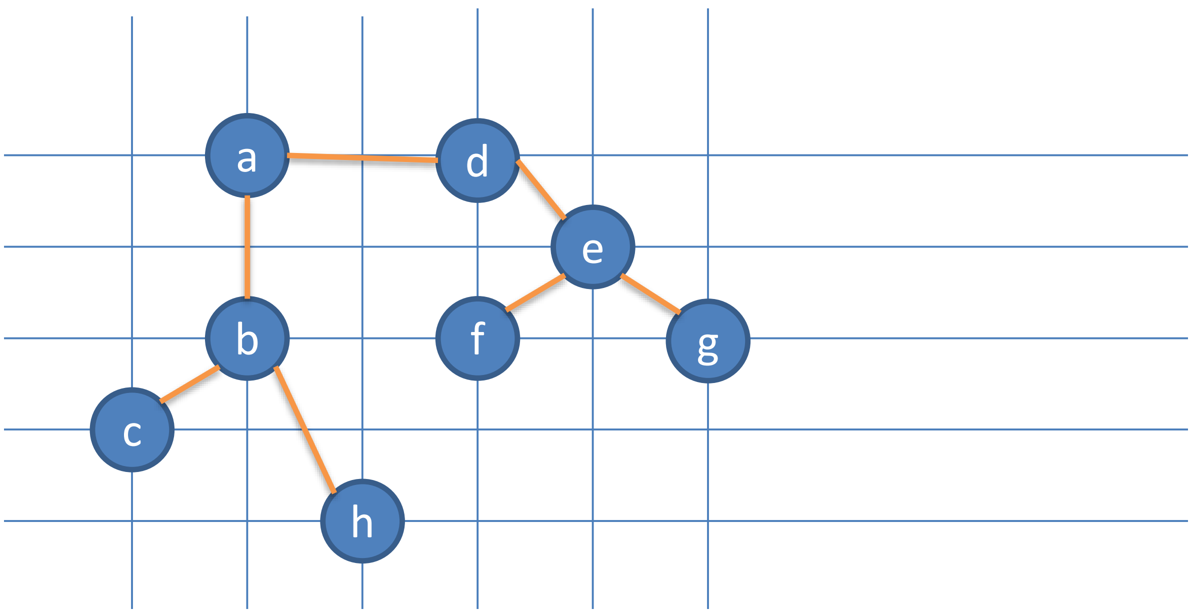


Use Prim's algorithm to get a MST

For any pair of vertices, there is a edge and the weight is the Euclidean distance

Triangle inequality is true, we can apply the approximation algorithm

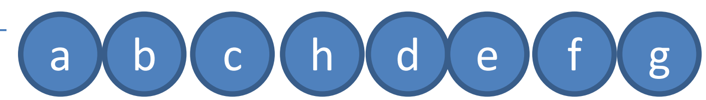
Traveling-salesman problem



Use Prim's algorithm to get a MST

Choose "a" as the root

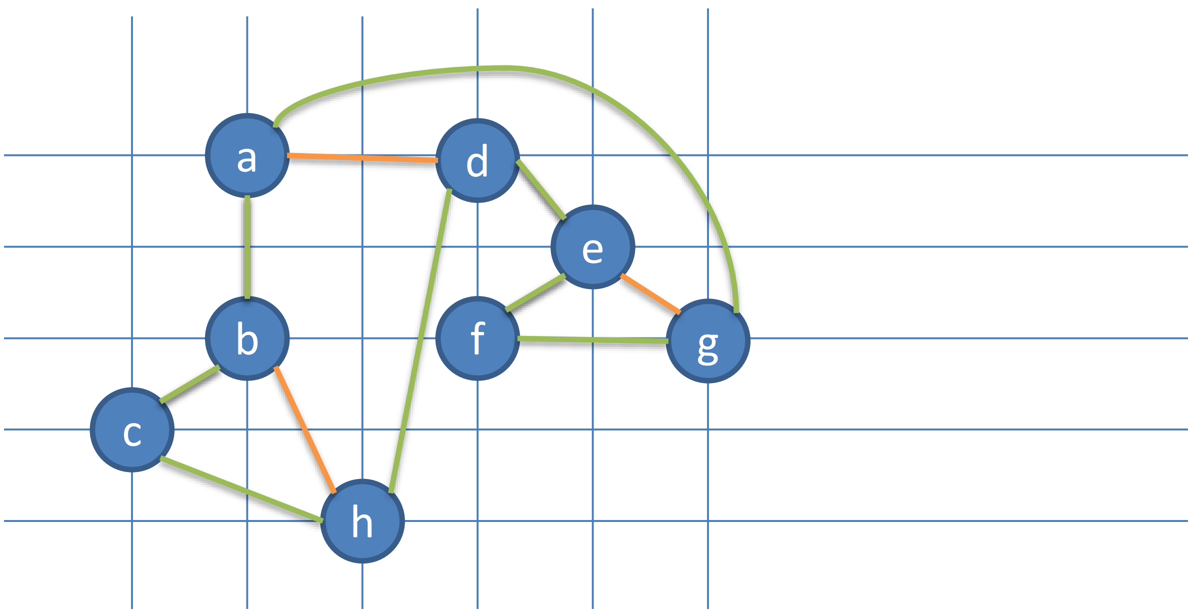
Preorder tree walk



For any pair of vertices, there is a edge and the weight is the Euclidean distance

Triangle inequality is true, we can apply the approximation algorithm

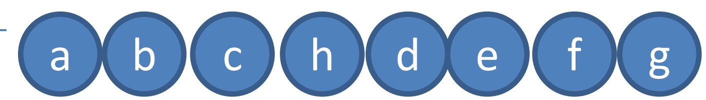
Traveling-salesman problem



Use Prim's algorithm to get a MST

Choose "a" as the root

Preorder tree walk



The route is then…

Because it is a 2-approximation algorithm

A TSP solution is found, and the total weight is at most twice as much as the optimal one

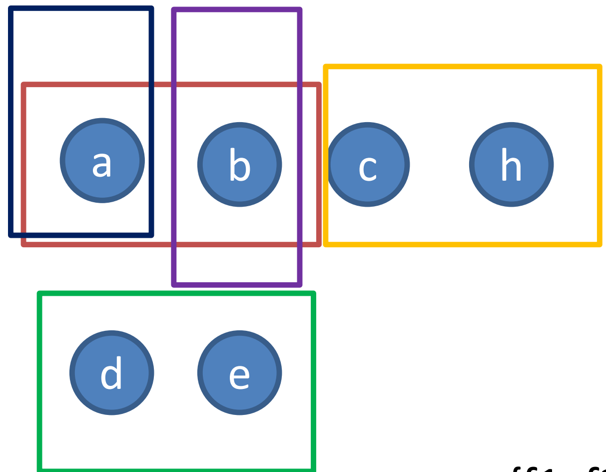
The set-covering problem

Set-covering problem

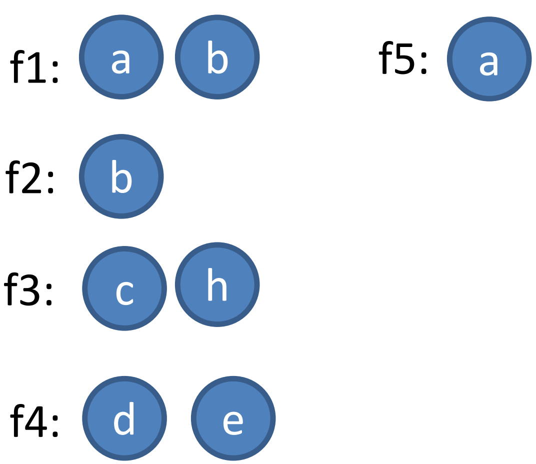
- Given a set X, and a family F of subsets of X, where F covers X, i.e. $\mathrm{X} = \bigcup_{S \in \mathrm{F}} S$.
- Find a subset of F that covers X and with minimum size

The set-covering problem

X:



F:



{f1, f3, f4} is a subset of F covering X

{f1, f2, f3, f4} is a subset of F covering X

{f2, f3, f4, f5} is a subset of F covering X

Here, {f1, f3, f4} is a minimum cover set

The set-covering problem

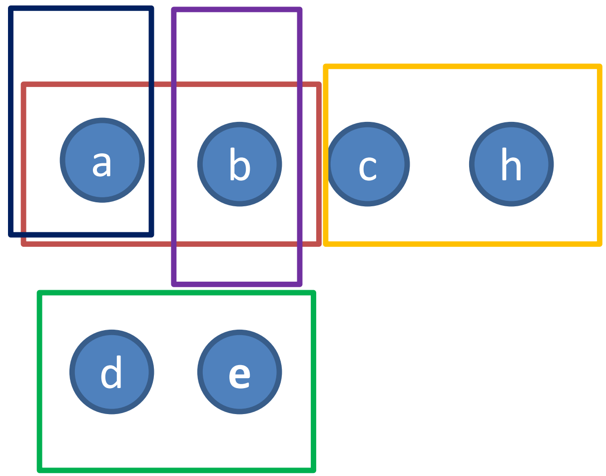
- **Set-covering problem** is NP-complete.
- If the size of the largest set in F is m, there is a $\sum_{i=1}^{m} 1/i$ - approximation polynomial time algorithm to solve it.

The set-covering problem

```
GREEDY-SET-COVER(X, F)
    U=X;
    C=∅;
    While(U≠ ∅){
        Select S∈F that maximizes |S∩U|;
        U=U-S;
        C=C∪{S};
    }
    return C;
```

The set-covering problem

X:



We can choose from f1, f3 and f4

Choose f1

We can choose from f3 and f4

Choose f3

We can choose from f4

Choose f4

F:

f1: a b

f2: b

f3: c h

f4: d e

f5: a

U: a b c h d e

C:

f1: a b

f3: c h

f4: d e

Set Cover and its generalizations and variants are fundamental problems with numerous applications. Examples include:

- selecting a small number of nodes in a network to store a file so that all nodes have a nearby copy,
- selecting a small number of sentences to be uttered to tune all features in a speech-recognition model [11],
- selecting a small number of telescope snapshots to be taken to capture light from all galaxies in the night sky,
- finding a short string having each string in a given set as a contiguous sub-string.

Summary

- P problems
- NP problems
- NP-complete problems
- NP-Hard problems
- The relation between P and NP
- Polynomial approximation algorithms