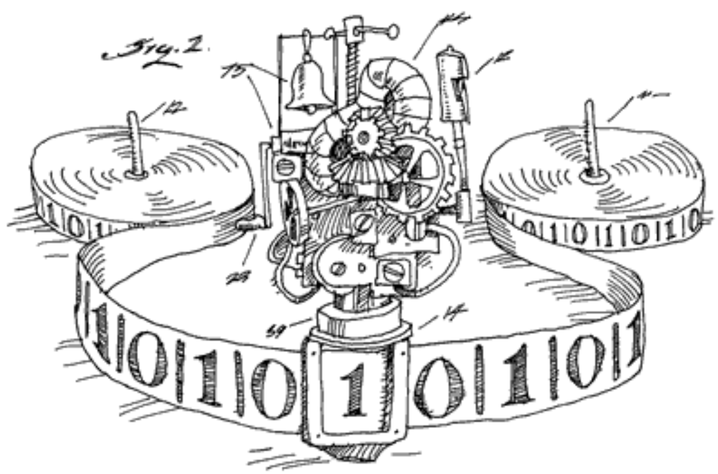
EECS 376: Foundations of Computer Science

Lecture 19 - Approximation Algorithms



What it means for an algorithm to be an α-approximation

Minimization Problems: **OPT** $\leq$ **ALG** $\leq$ α ▪ **OPT,** $\alpha \geq$ **1** (smaller α is better)

Maximization Problems: **OPT** $\geq$ **ALG** $\geq$ α ▪ **OPT,** $\alpha \leq$ **1** (larger α is better)

ALG = value returned by our algorithm

OPT = Optimal value

α is the *approximation ratio*

Two ingredients in approximation analysis

Minimization Problems:

- **Upper bound on ALG**
- **Lower bound on OPT**

Maximization Problems:

- **Lower bound on ALG**
- **Upper bound on OPT**

Next: an approximation algorithm for the problem of Maximum Cut…

Max Cut

Graph Cuts

def ①

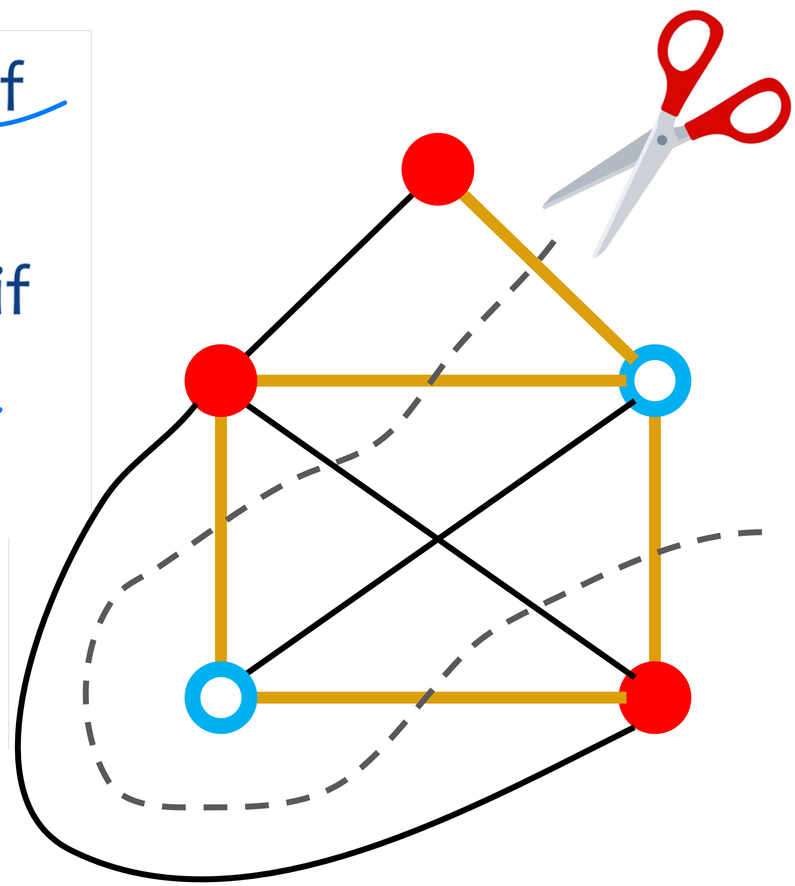
* A ***cut*** of a graph is a *partition* of its vertices $(S, \bar{S})$.

def ②

* An edge ***crosses*** the cut $(S, \bar{S})$ if one of its endpoints is in S and the other is in $\bar{S}$.

def ③

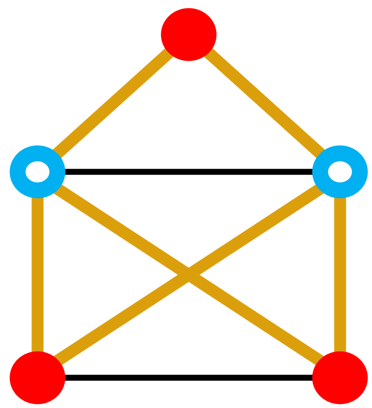
* The ***size*** of a cut $(S, \bar{S})$ is the number of edges crossing it.



Maximum Cut Problem

Max-Cut Problem: Given a graph, find a cut of maximum size.

- decision version is NP-complete (we won't prove)

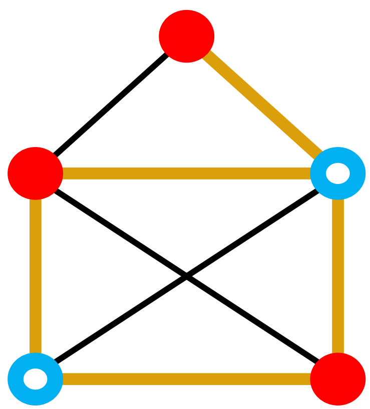


Has applications in network/circuit design, physics, and more…

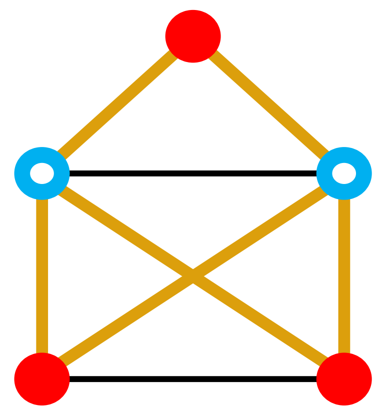
Approximate Maximum Cut

We will show a poly-time ½-approximation

(i.e. the cut returned by our algorithm is at least ½ the size of a max cut)



is a 1/2-approx. of optimum:



Technique: Local Search

Idea:

- Start with an arbitrary cut.
- Pick a vertex v
- Switch the side of v if it increase the cut size *(this is a local search)*.

Quiz:

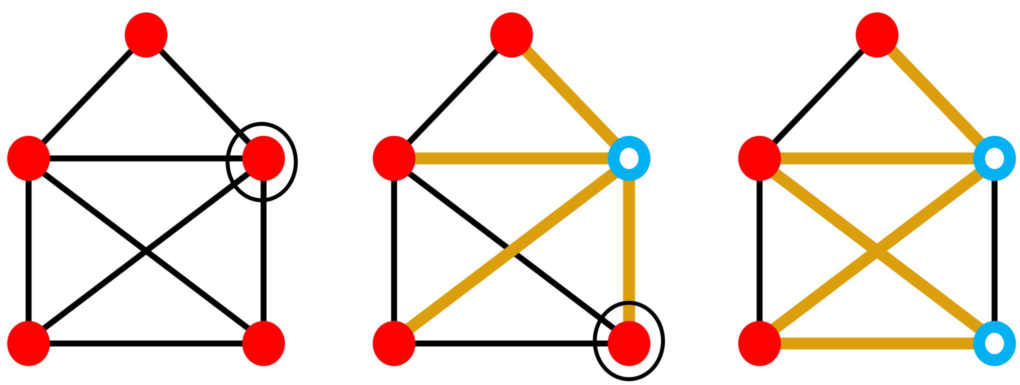
When switching the side of v will increase the cut size?

Ans:

#neighbors of v on the same side > #neighbors of v on the other side

Algorithm

- Start with an arbitrary cut.
- While there is a vertex v such that
  #neighbors of v on the same side > #neighbors of v on the other side
  - Switch the side of v
- Return the cut.



Analysis: Running Time

Why does the algorithm terminate?

因为 vertex 有限

Why is it polynomial time?

O(|V|)

Analysis: Approximation Ratio

ALG : #edges in our cut
OPT : #edges in an optimal cut

Want to show, ALG $\geq m/2$. **Lower bound on ALG**

OPT $\leq m$. **Upper bound on OPT**

$\Rightarrow$ **ALG** $\geq \frac{1}{2} \cdot$ **OPT**

- **OPT** $\leq$ m is clear.
- Why **ALG** $\geq m/2$?

(actually OPT $\geq \frac{m}{2}$)

Analysis: Approximation Ratio

ALG : #edges in our cut
OPT : #edges in an optimal cut

Want to show, ALG ≥ m/2. | Lower bound on ALG

OPT ≤ m. | Upper bound on OPT

⇒ ALG ≥ ½ ⋅ OPT

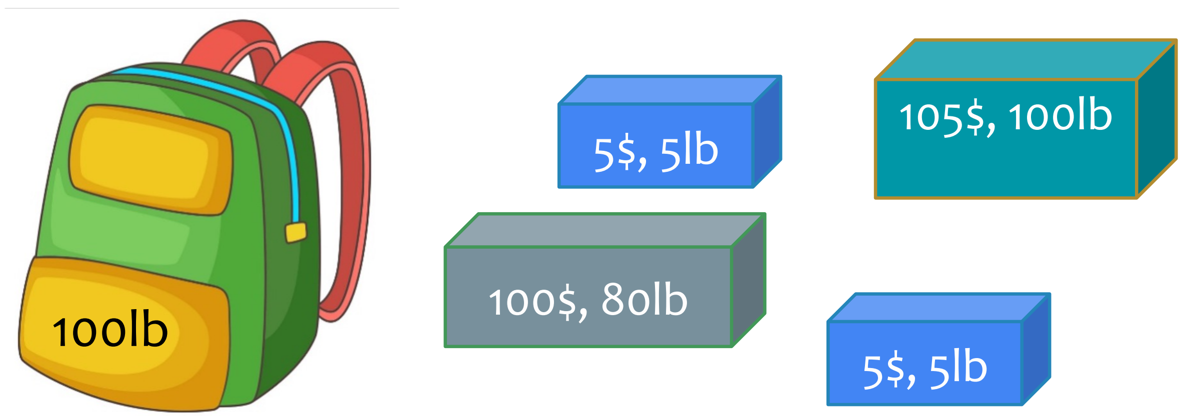
- OPT ≤ m is clear.
- Why ALG ≥ m/2?

$$\text{ALG} = \frac{1}{2}\Sigma_v(\#v\text{'s incident cut edges}) = \frac{1}{2}\Sigma_v \frac{\deg(v)}{2} \geq \frac{m}{2}$$

Knapsack

Knapsack Problem

Given a backpack with weight capacity **W,** and a set of **n** items each with an integer value $\mathbf{v_i} \leq \mathbf{V}$ and weight $\mathbf{w_i} \leq \mathbf{W}$, what is the largest total value of a set of items that fit in the backpack (i.e. total weight of set ≤ **W**)?

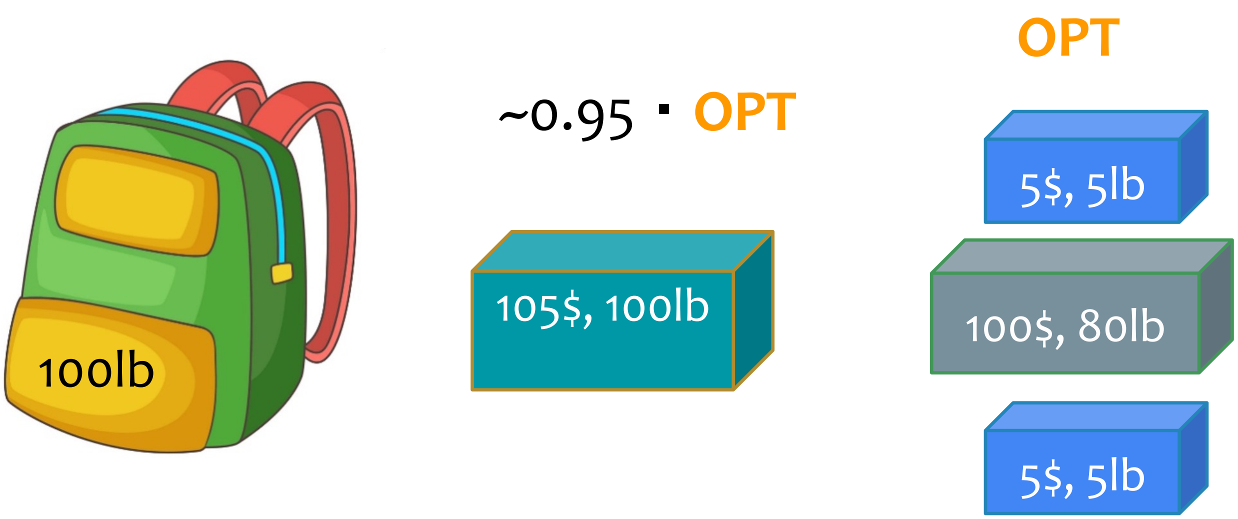


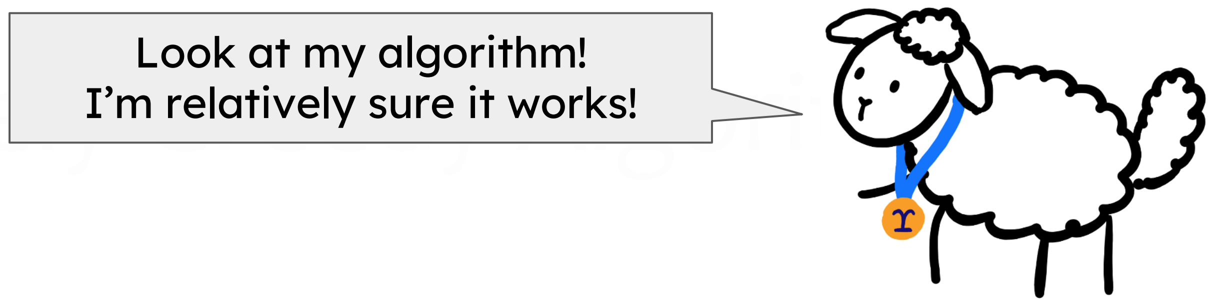
On the HW: **Knapsack is NP-hard**

Approximate Knapsack

We will show a poly-time ½-approximation

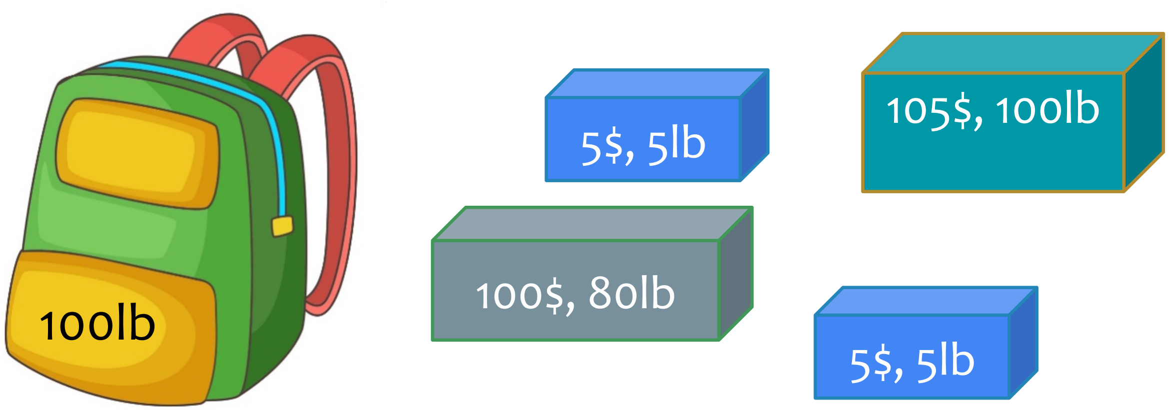
i.e. the total value of the items chosen by our algorithm is at least ½ the optimal value.

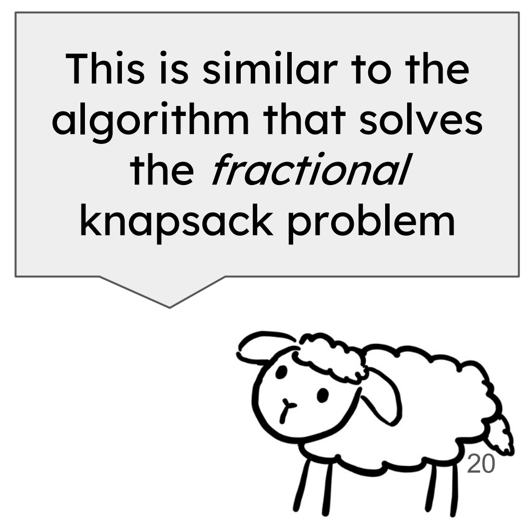




Relatively-Greedy Algorithm:

- Consider items in decreasing order by *relative* value (breaking ties arbitrarily) i.e. the ratio **value/weight**
- Greedily select item if it fits in remaining capacity.





Your task: How bad is the approximation ratio of the Relatively-Greedy algorithm?

Construct an example to support your claim.

(An example consists of: weight of backpack, and weight/value of each item)

100 lb

2\$, 2lb

99\$, 100lb

frac = 1

frac = 0.99

ALG = 2, OPT = 99

$\alpha = 0.$

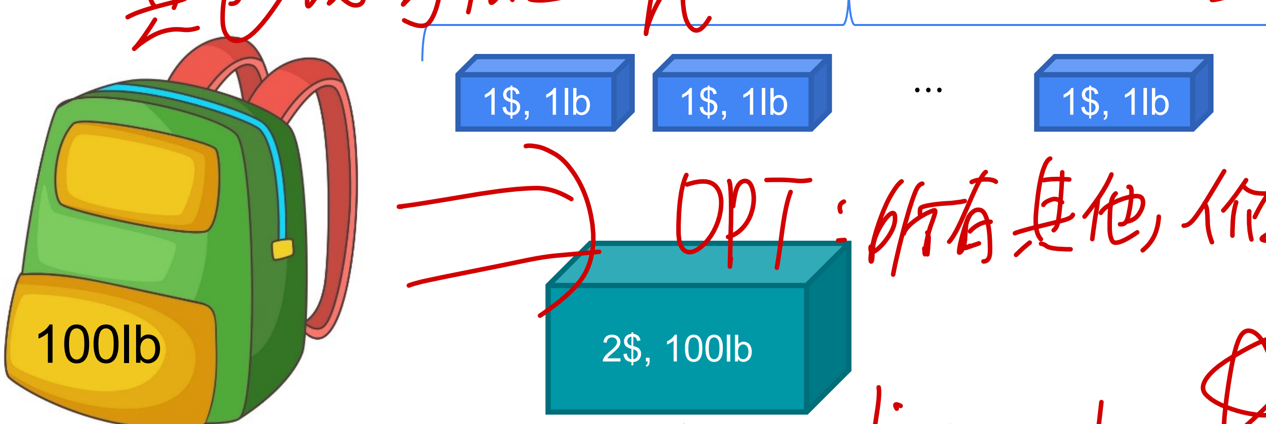
Single-Greedy Algorithm: Take the one single item of largest value that fits in the backpack. (Don't take any more items.)

n个 item，总价值 V，总重 W，承重 $\frac{W}{2}$

Example to show approximation ratio is bad:

1个价值 $\frac{V}{n}$ + a bit，重 $\frac{W}{2}$

其它均匀价值 $\frac{V}{n}$ − a bit，合重 $\frac{W}{2}$



OPT：所有其他，价值 $\frac{n-1}{n}V$

ALG $\approx \frac{V}{n}$. $\alpha = \lim_{n\to\infty} \frac{1}{n-1} = 0$

Combined-Greedy Algorithm:

- Run **Relatively-Greedy** and **Single-Greedy**
- Take the best of the two solutions

One can show: Combined-Greedy is a ½-approximation!

Example of combined algorithms in practice: ***The Netflix Challenge (2009)***
"[The winning team] simply ran hundreds of algorithms from their 30-plus members and combined their results into a single set, using a variation of weighted averaging that favored the more accurate algorithms."

Metric Traveling Salesman Problem

Approximate Metric-TSP

Input:

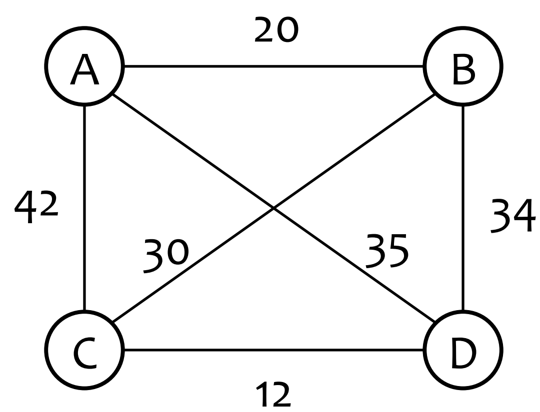
a **complete graph** with n vertices.

Edge weights form a **metric,** i.e., they obey the **triangle inequality:**

for any x, y, z $dist(x,z) \leq dist(x,y) + dist(y,z)$

Output:

What is the minimum length cycle visiting each vertex once?



The decision version of Metric-TSP is NP-complete.

We will show a poly-time 2-approximation.

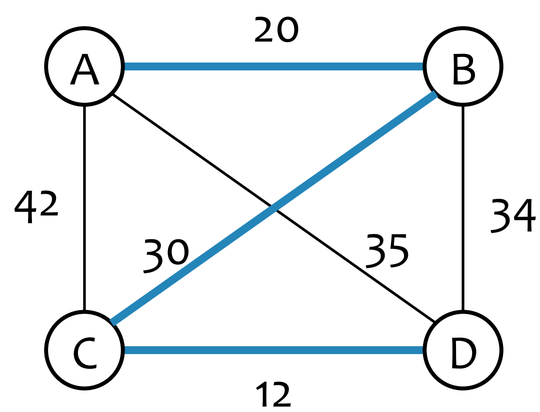
(returns a tour of length at most 2 times the optimal tour)

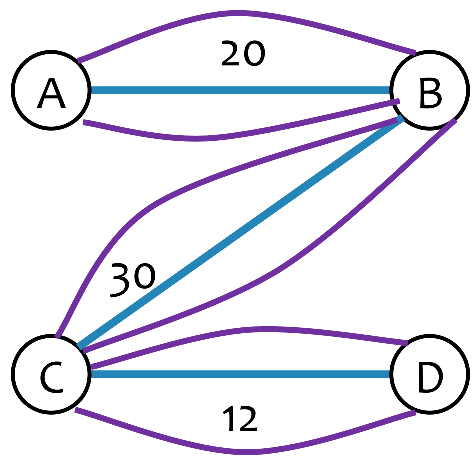
Algorithm

Step 1: Find an **MST** (in polynomial time)

Step 2: Walk around the perimeter of the **MST** to form **"tree-tour"**

tree-tour is not a legitimate TSP tour!





tree-tour = 2·MST

Algorithm

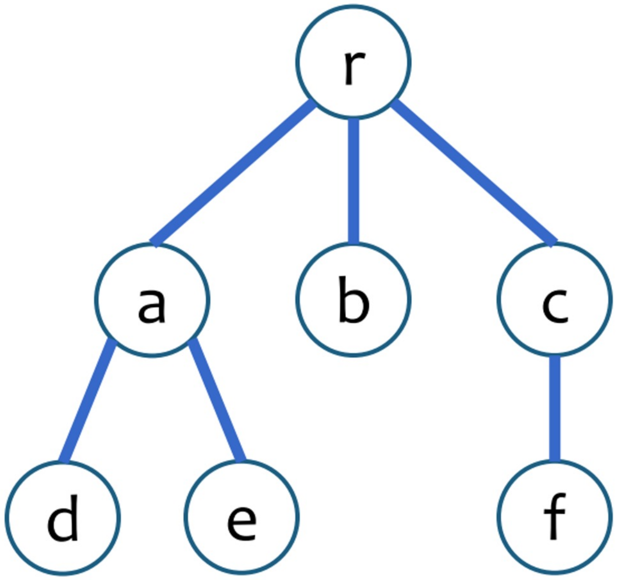
Step 1: Find an **MST** (in polynomial time)

Step 2: Walk around the perimeter of the **MST** to form **"tree-tour"**

tree-tour is not a legitimate TSP tour!

If you wanted to code it:

```
Find-Tour(u)
   Let v_1, … , v_k be u's children
   For i = 1, … , k
      T = T + (u, v_i)
      Find-Tour(v_i)
      T = T + (v_i, u)
```



tree-tour = 2·MST

Algorithm

Step 1: Find an **MST** (in polynomial time)

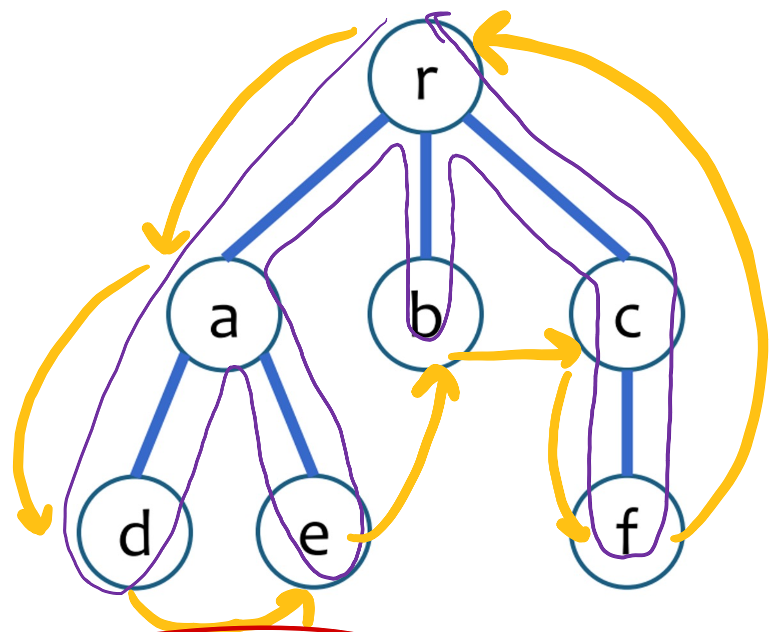
Step 2: Walk around the perimeter of the **MST** to form **"tree-tour"**

tree-tour is not a legitimate TSP tour!

走两遍(来回)

Step 3: **"Shortcut"** tree-tour

- Repeatedly visit
  the next unvisited vertex in tree-tour



Why shortcut tree-tour ≤ **tree-tour** ?

- **Triangle inequality!**

ALG ≤ **tree-tour** = **2·MST**

Analysis

We have shown

ALG ≤ **tree-tour** = 2· **MST**

To get 2-approximation, ALG ≤ 2 · OPT, we will show

$\boxed{\text{OPT} \geq \text{MST}} \geq \frac{1}{2} \text{ALG}$

Why is this?

- An optimal cycle has weight at least that of some spanning tree

Can we do better than a 2-approximation?

Yes!

* [Christofides 1976] 1.5-approximation
* [Karlin-Klein-Oveis Gharan 2021] $(1.5 - 10^{-36})$-approximation.
* [Karpinski-Lampis-Schmied 2013] No 1.008-approximation unless P = NP.

https://en.wikipedia.org/wiki/Christofides_algorithm

https://dl.acm.org/doi/10.1145/3406325.3451009

Wrap Up

Ways to deal with NP-Hardness

1. **Approximation algorithms**
2. Restrict to **special classes of inputs**
   - randomly-generated inputs, planar graphs, …
   - **Fixed-parameterized algorithms**
3. **Heuristics:** algorithms without provable guarantees that seem to work well in practice
   - SAT solvers sometimes do well in practice
4. If your **input is small**, sometimes you can afford to run an exponential-time algorithm

Goodbye Complexity…

EECS 574: Computational Complexity

The Netflix Challenge (2009)

- more complexity classes
- hardness of approximation

EECS 477: Introduction to Algorithms

- more NP-hardness proofs
- more approximation algorithms