Ticket to Ride: Steiner Tree

Patrick Schäfer
patrick.schaefer@hu-berlin.de

Semesterprojekt: Implementierung eines Brettspiels, WS 18/19

Agenda

- Today, 13:15
  - Competition
  - Short talk on Steiner Tree Approximation

Questions

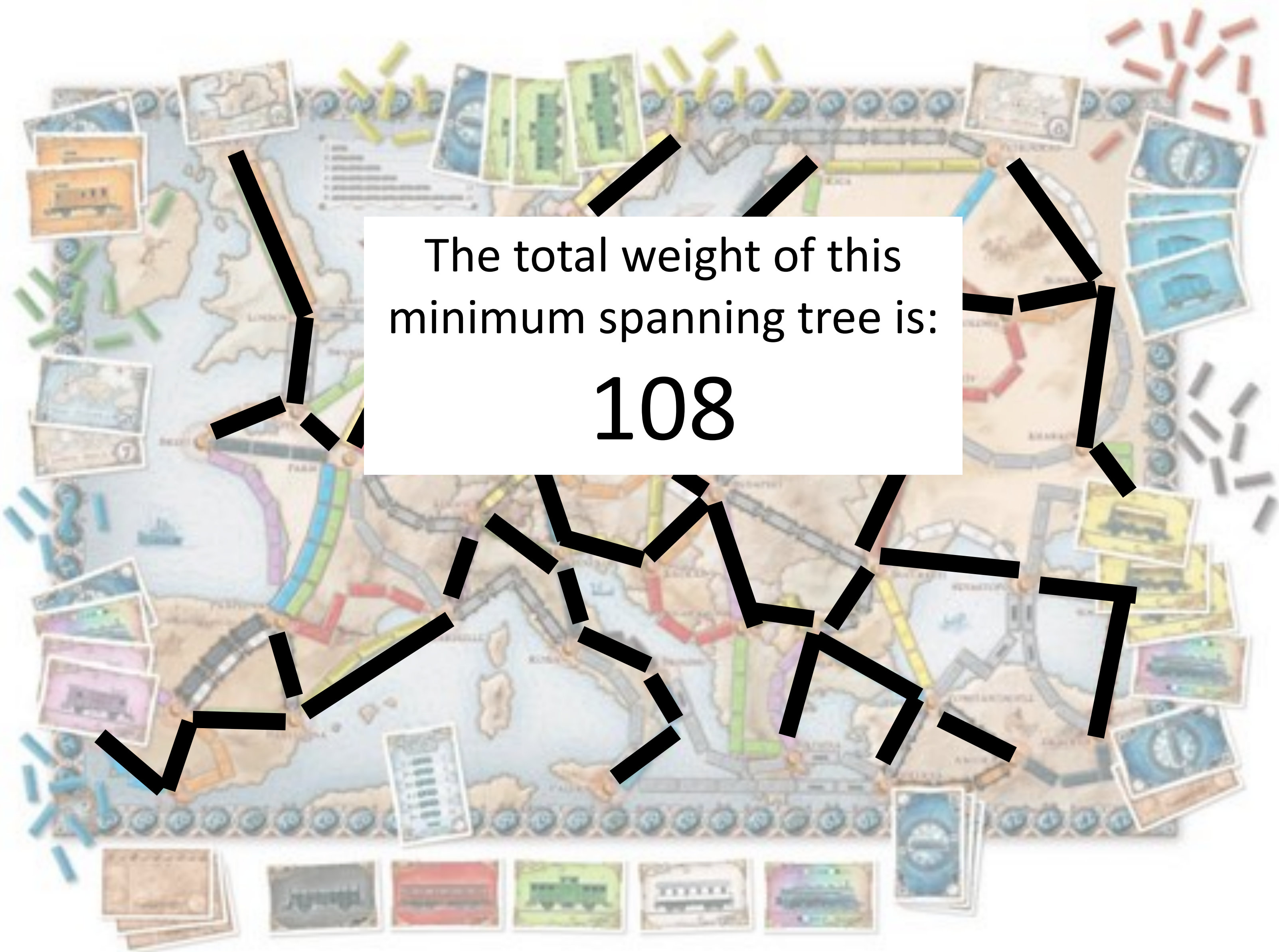
- Are there any questions related to…
  - TTR-Server
  - TTR-Protocol
  - C#-Client-Implementation

- Benchmark AI and Shell-Script (also linked on the webpage):
  - https://box.hu-berlin.de/f/5b76e7c0f9084980ac63/?dl=1

Ticket to Ride and Graph Theory

- G = (V,E), V=Cities, E = Railways.
- Each vertex of the graph represents one city in Europe
- An edge connects two cities
- Each edge has a color and a length (cost)
- The graph contains more edges than any player can claim

A MST for Ticket to Ride



The total weight of this minimum spanning tree is:

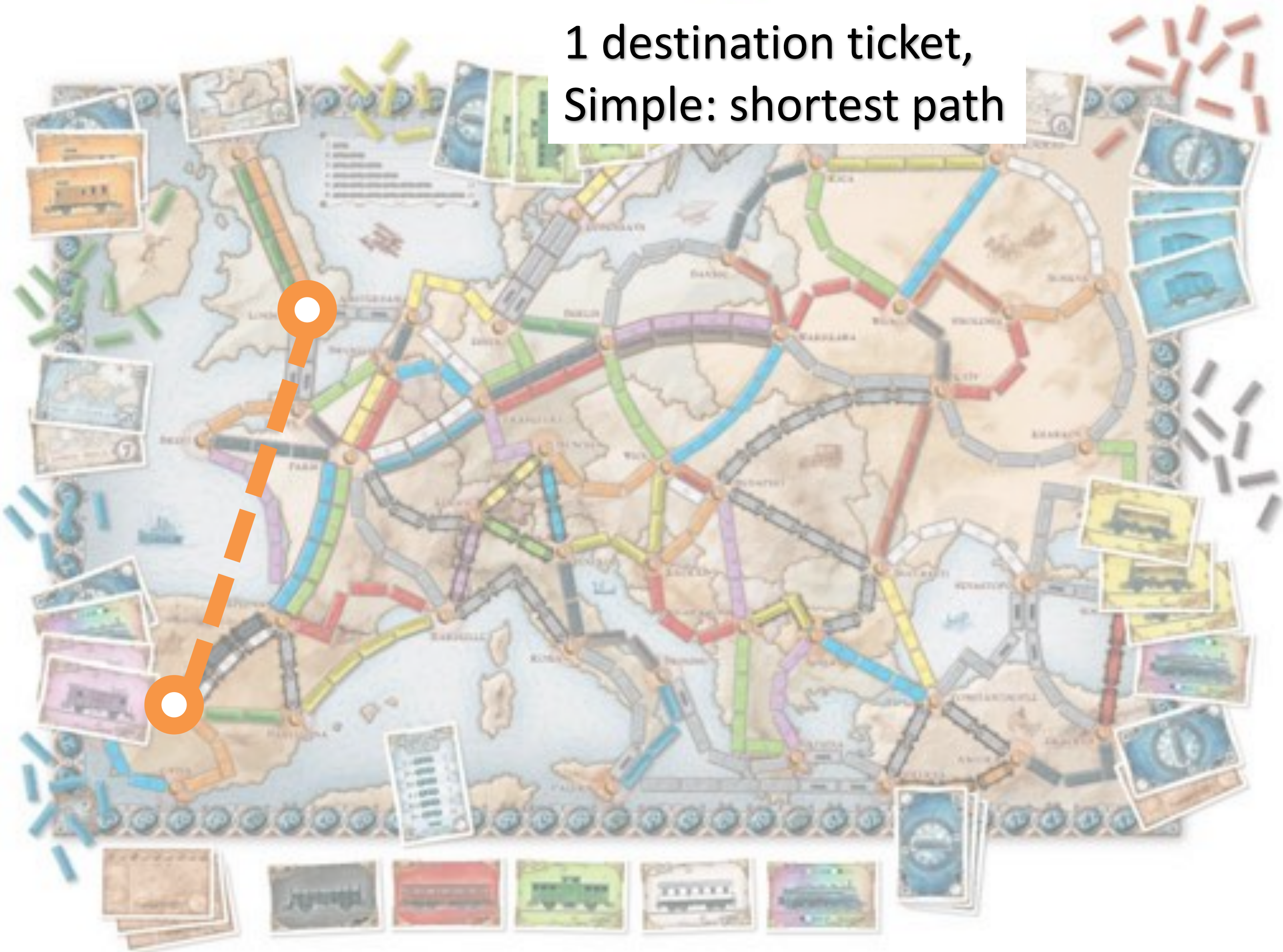
108

Minimum Spanning Tree (Forest)

- A spanning tree of the full graph would guarantee that any destination ticket is fulfilled.
- But payers do not have enough train tokens to claim a spanning tree of the full graph (45 vs 108).
- Thus, the best strategy is to capture a minimum spanning tree or forest of a subset of vertices (based on the destination tickets).

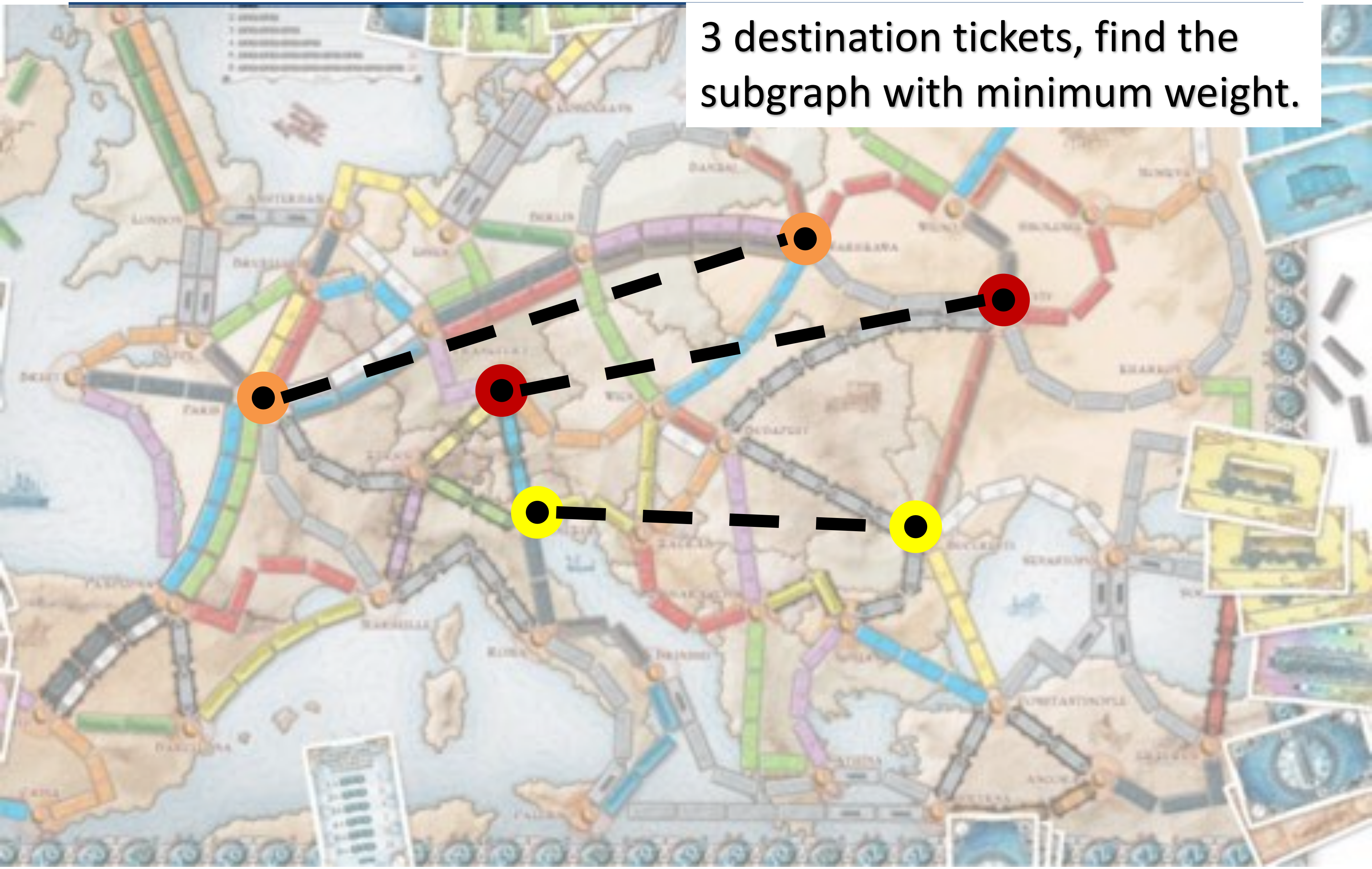
- **Steiner Tree / Forest:** Given an undirected, weighted graph G=(V,E) and a subset of vertices V’, referred to as terminals, we search the subgraph G’ with minimum weight, that connects all terminals (and may include additional vertices).

Special Case: 2 terminals



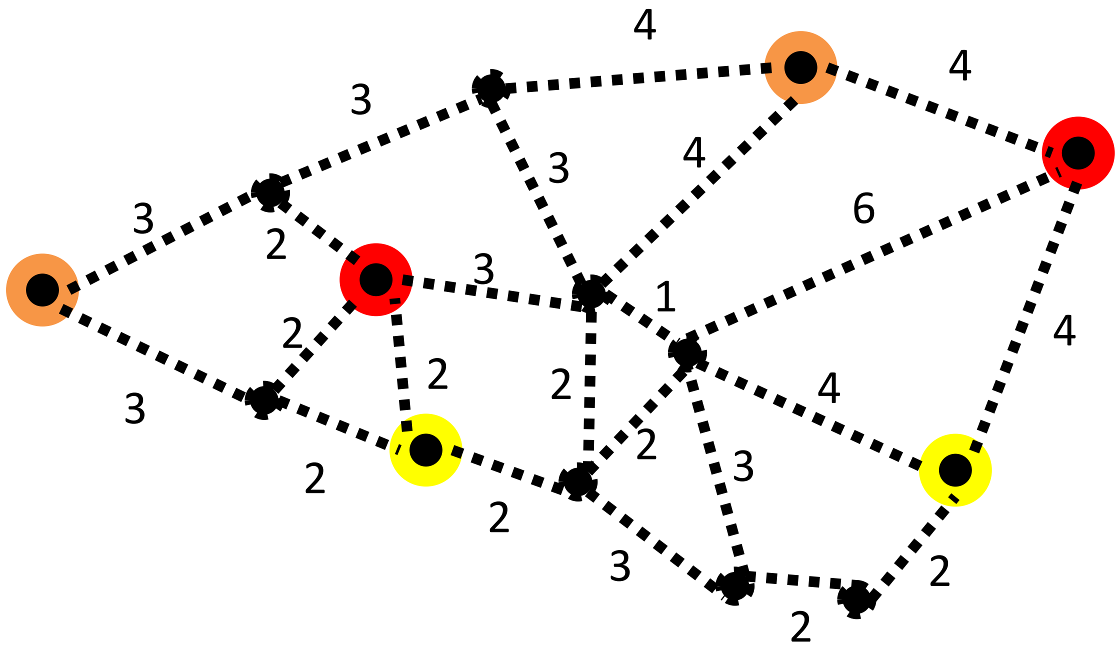
Minimum-Weight Subtree

3 destination tickets, find the subgraph with minimum weight.

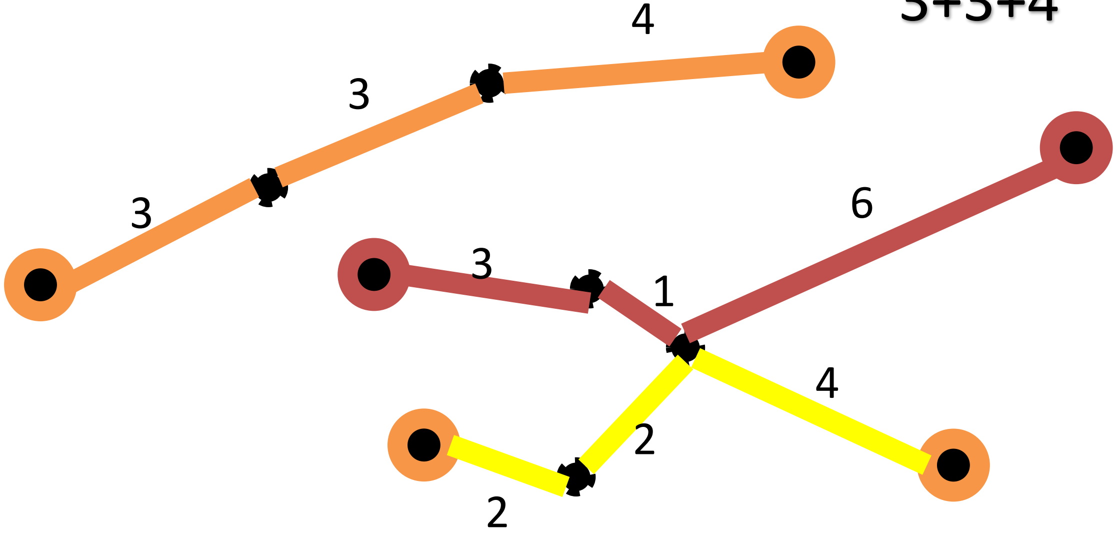


Minimum-Weight Subtree on Destination Tickets

3 destination tickets, find the subgraph with minimum weight.



Dijkstra



3+3+4

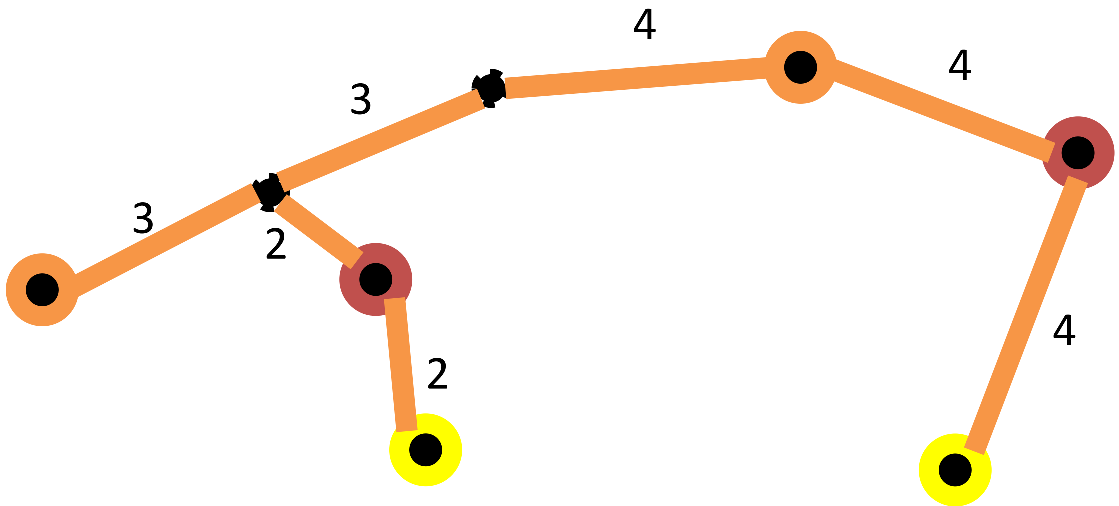
3+1+6

2+2+4

Total : 10+10+9=29

Minimum-Weight Subtree on Destination Tickets

One cost-optimal Steiner tree



Total : 3+3+4+4+4+2+2=22

Approximate Algorithm

- Steiner Tree optimization problem is NP-hard, thus there is likely to be no exact polynomial time algorithm.
- There are *heuristic* algorithms with polynomial time, that have upper bound guarantees on the maximum cost.
- Implementing a good algorithm helps greatly for the AI-Challenge.

Steiner Tree Special Cases

- |V’|=2: Shortest Path
- |V’|=V: Compute the minimum spanning tree (MST)

- Idea: if we had a fully connected graph, then we can get the optimal solution using MST
  - Add edges to G’ that represent the shortest path between all nodes

Approximation Algorithm I

- G=(V,E), *cost* are the edge costs, S is a set of terminals
- Construct graph G'=(S,E'), cost'

  build a fully connected graph:

  for any pair of vertices cost'=distance of the shortest path

  compute T = MST(G')

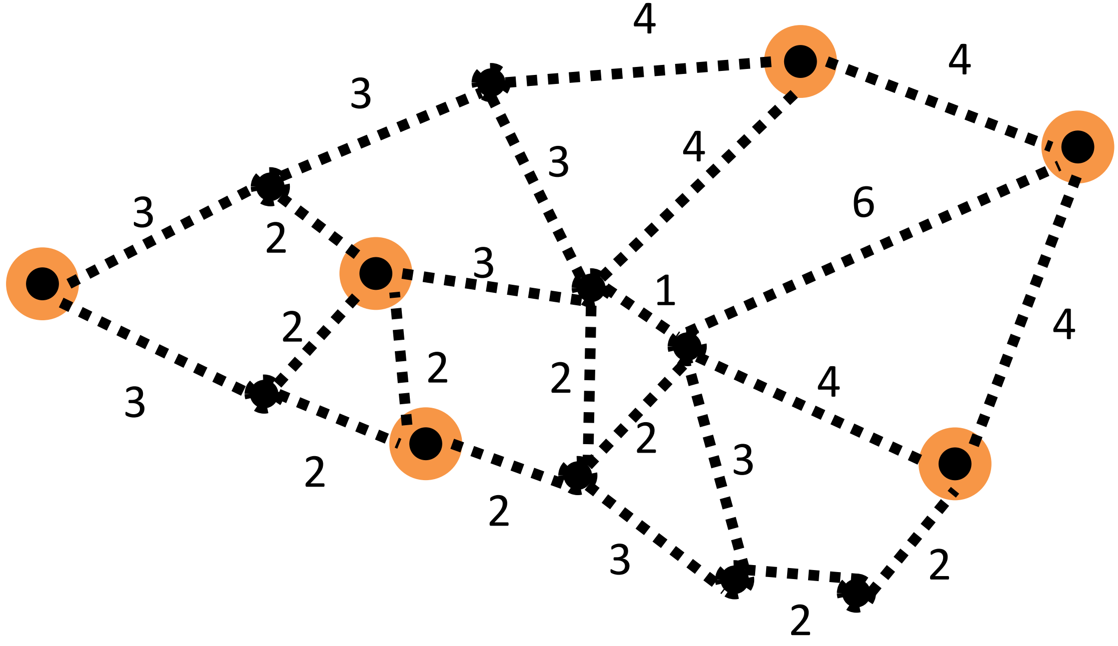
- The minimum spanning tree T contains edges that represent shortest paths

- Recover Steiner Tree from T

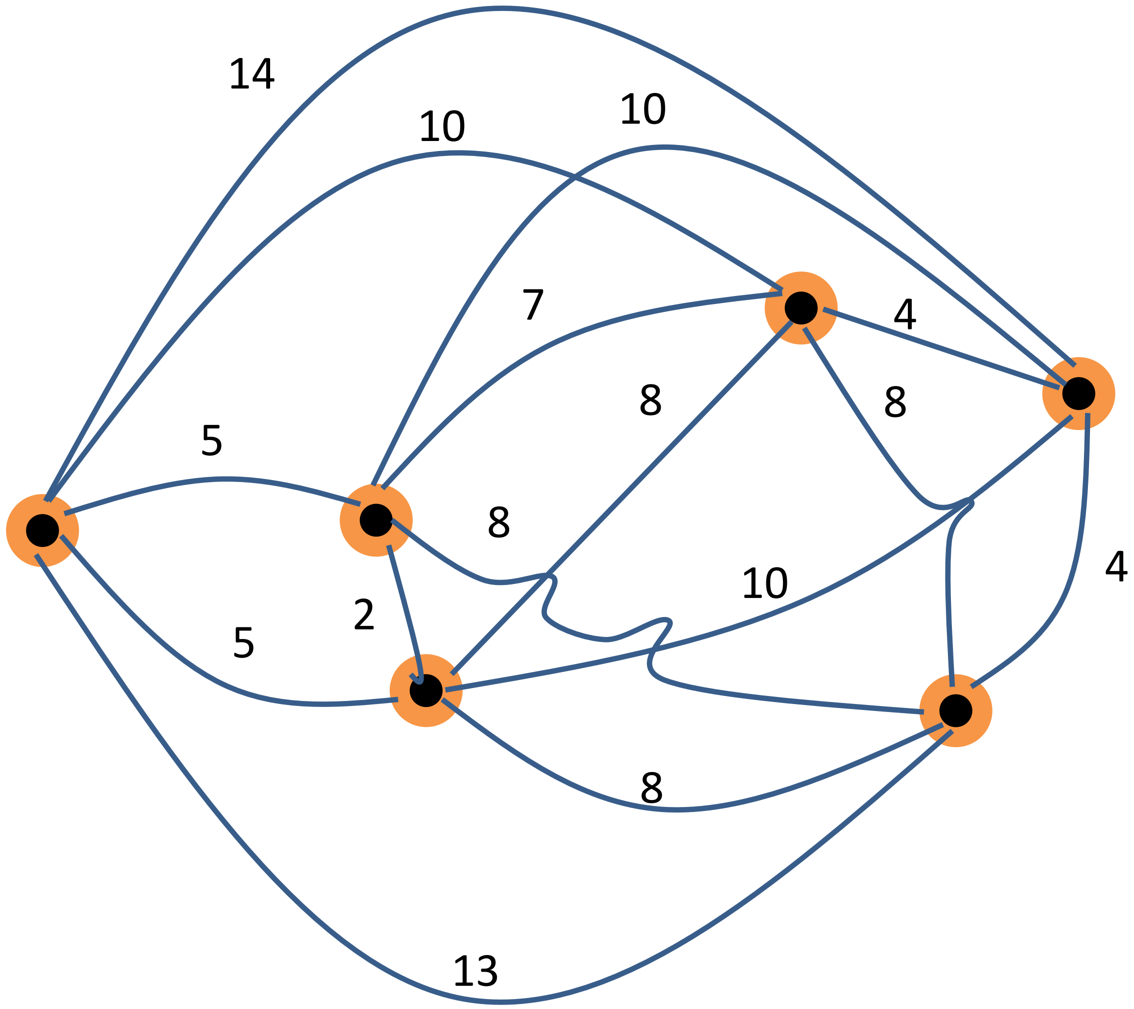
  T* = recover shortest paths in G that correspond to edges in T

  if present, remove cycles from T*

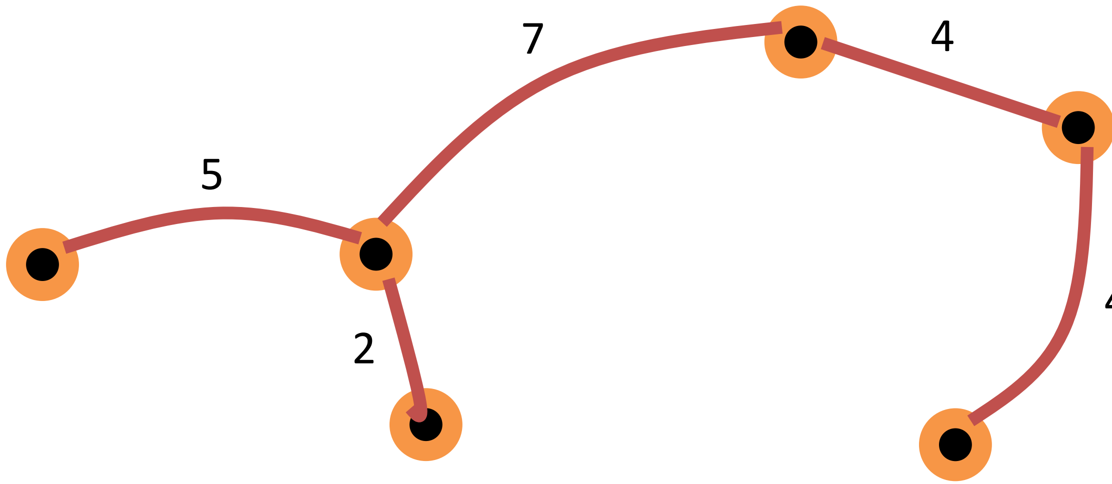
The graph G



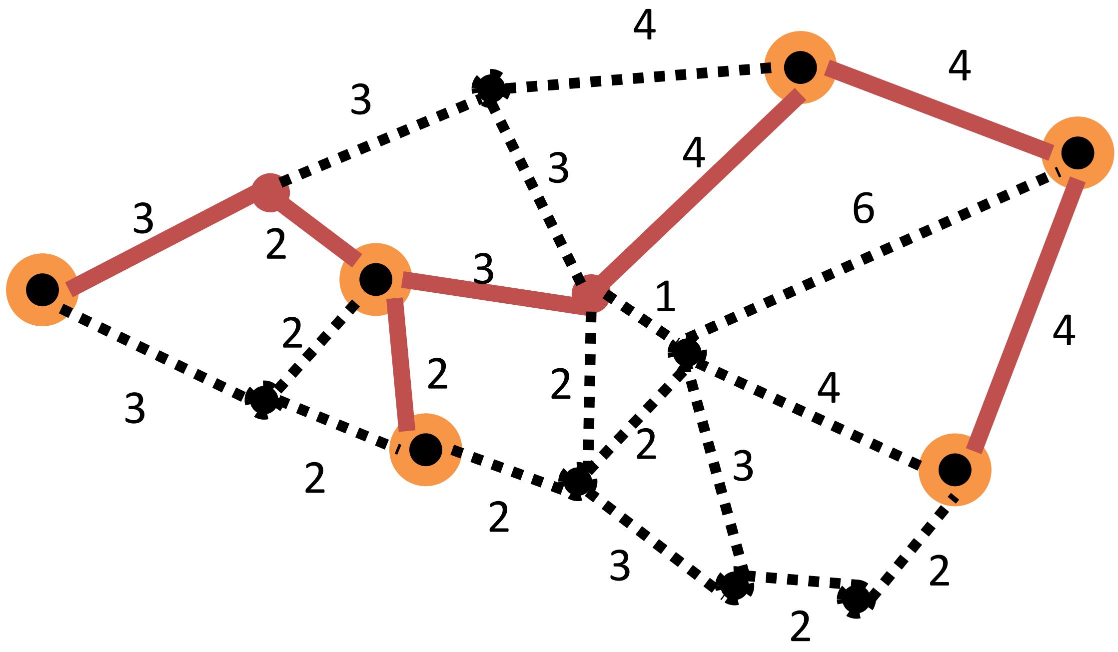
Build a fully-connected graph G'



A minimum spanning tree on G’



2-approximate Steiner tree



Approximation algorithm

- A α-approximation algorithm is a polynomial-time algorithm that returns a solution of cost at most α times the cost of an optimal solution

2-approximation

- 2 x optimal costs

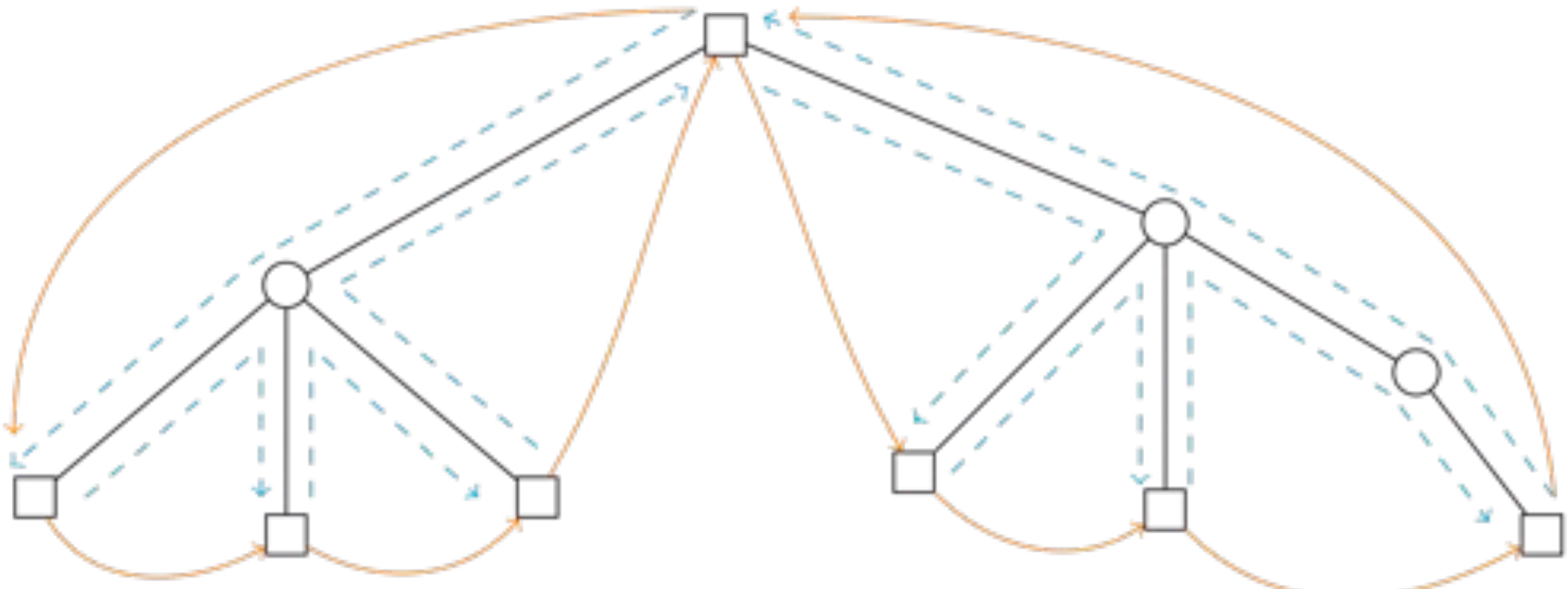
  = cost a DFS traversal (in blue) on the optimal tree

  that visits every edge exactly twice

  ≥ cost of some spanning tree constructed on shortest

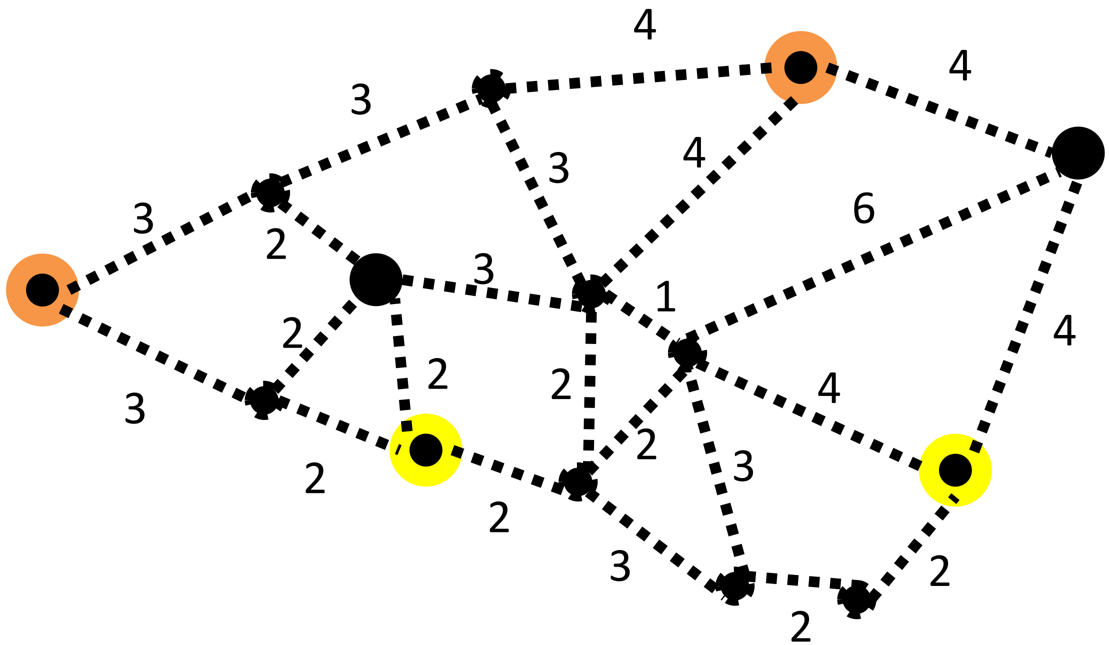
  paths (orange)

  ≥ cost of the MST on orange edges

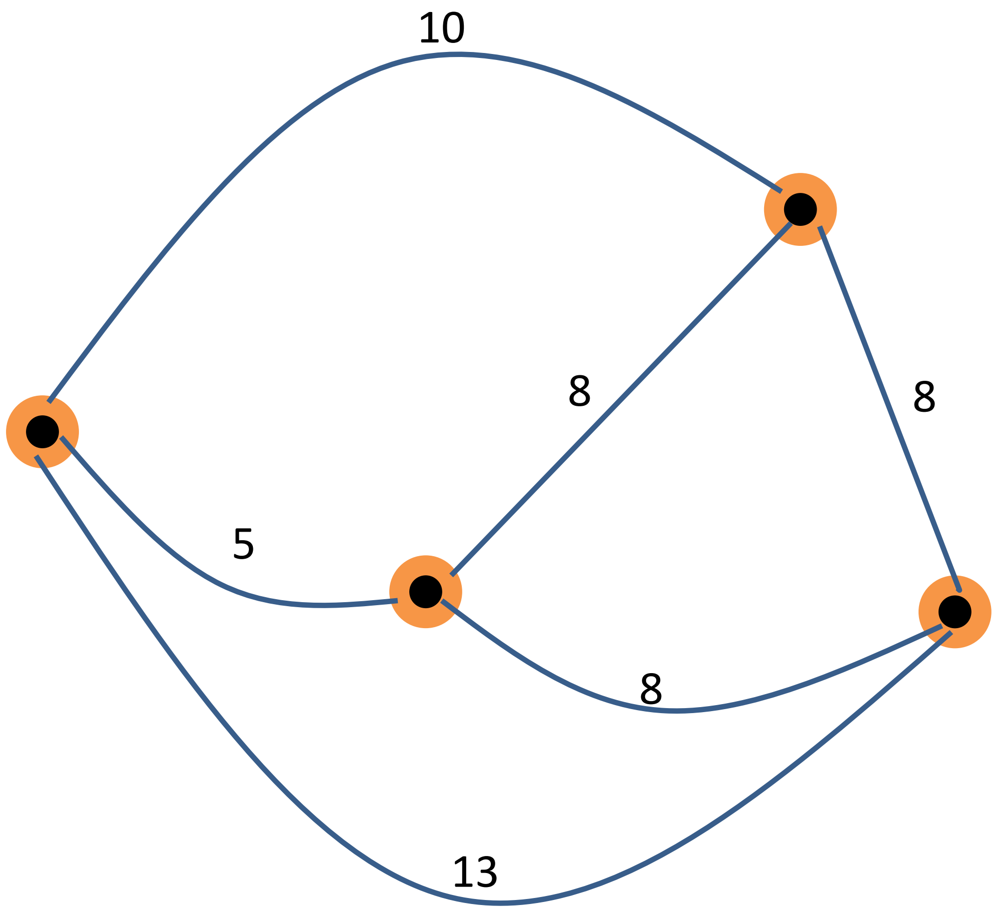


https://services.math.duke.edu/~yh89/other/Lecture16-17.pdf

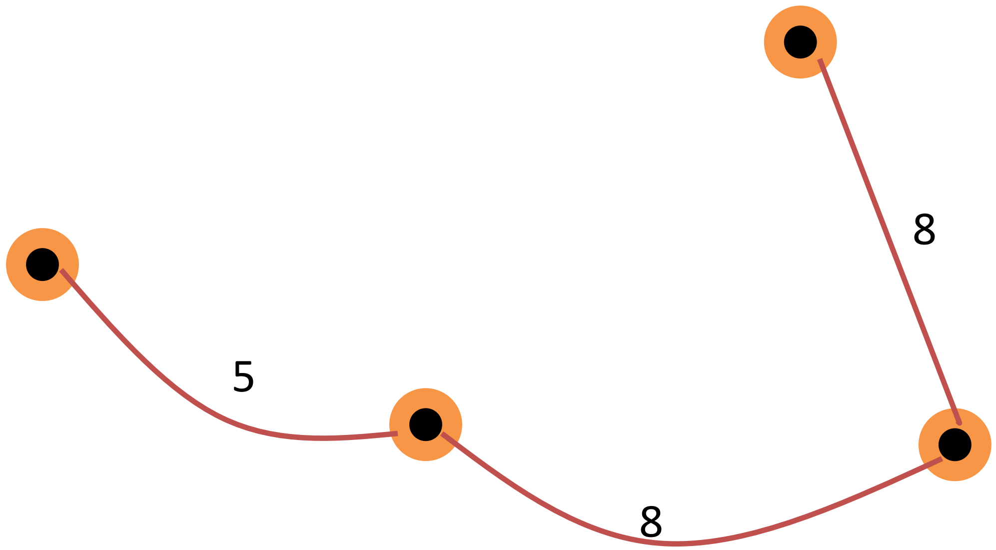
Minimum-Weight Subtree on Destination Tickets



Build a fully-connected graph G'

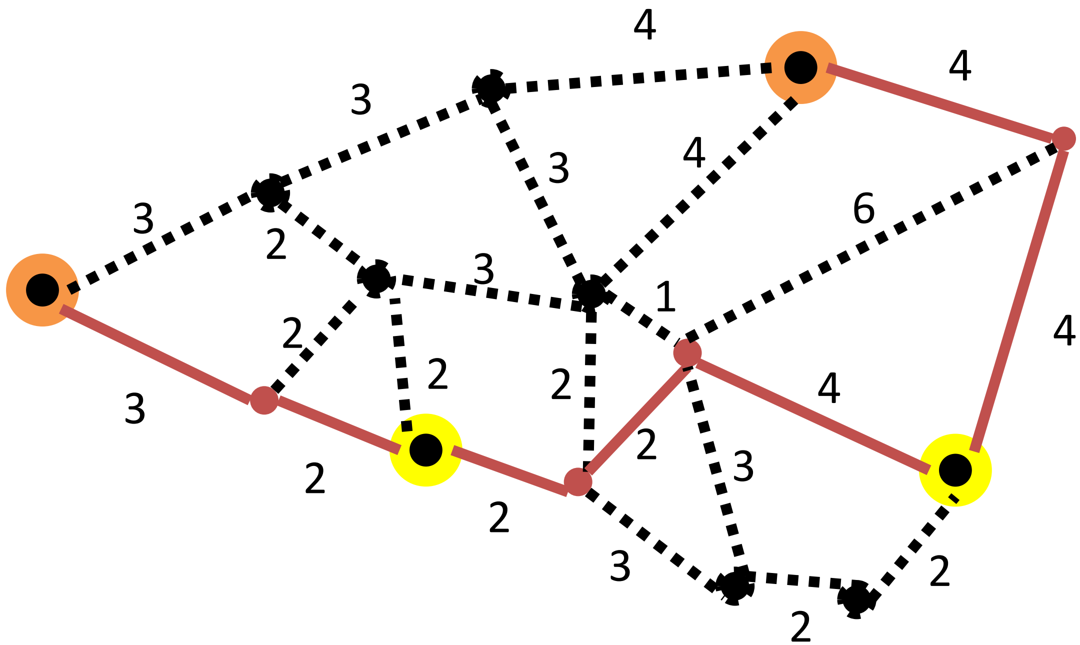


A minimum spanning tree on G’



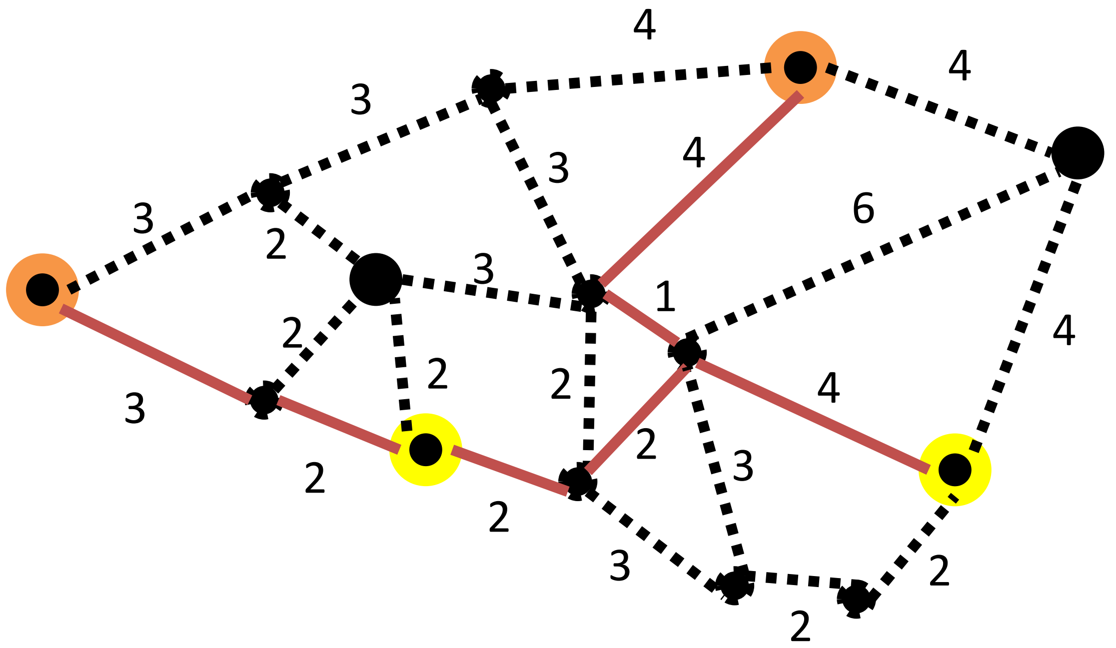
Cost = 21

2-approximate Steiner tree



Cost = 5+8+8=21

An optimal (?) Steiner tree



Cost = 5+9+4=18

An alternative Approximation Algorithm

- G=(V,E), S is a set of terminals
- Heuristic for Steiner tree

  start with subset T consisting of one terminal

  while T does not span all terminals:

  select a terminal t not in T that is closest to a vertex in T

  add to T the shortest path that connects t with T

- Improve on solution

  build a subgraph of G=(V,E) induced by the vertices in found solution T

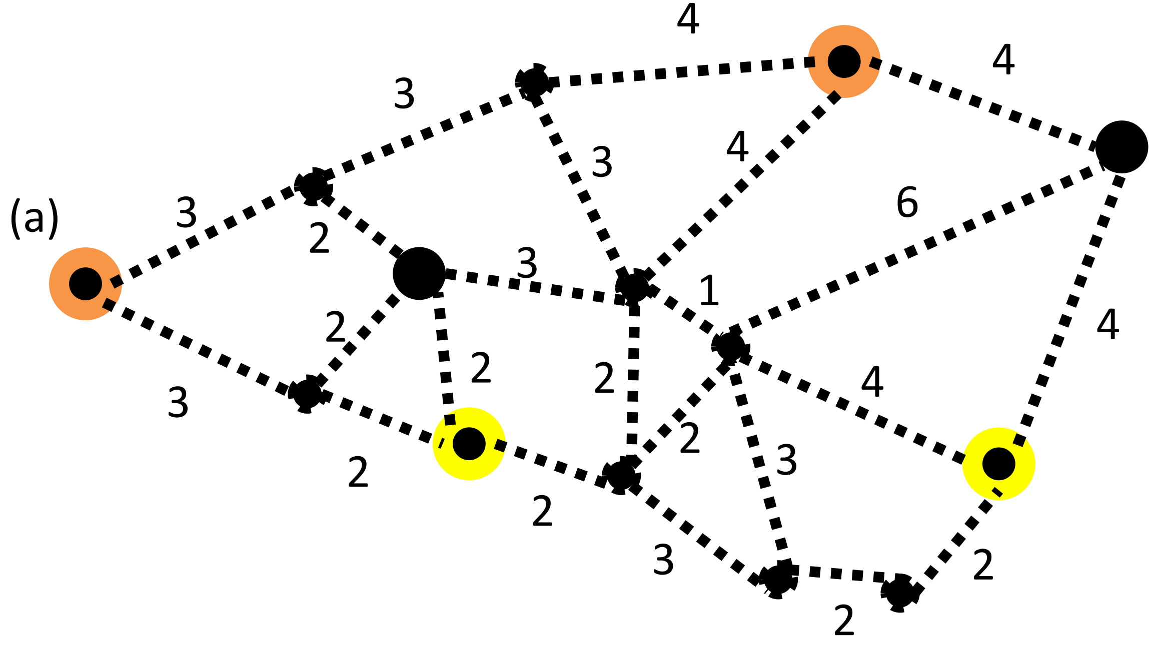
  compute a MST

  repeat until there are no non-terminal leaves

  delete non-terminals that are leaves of the MST

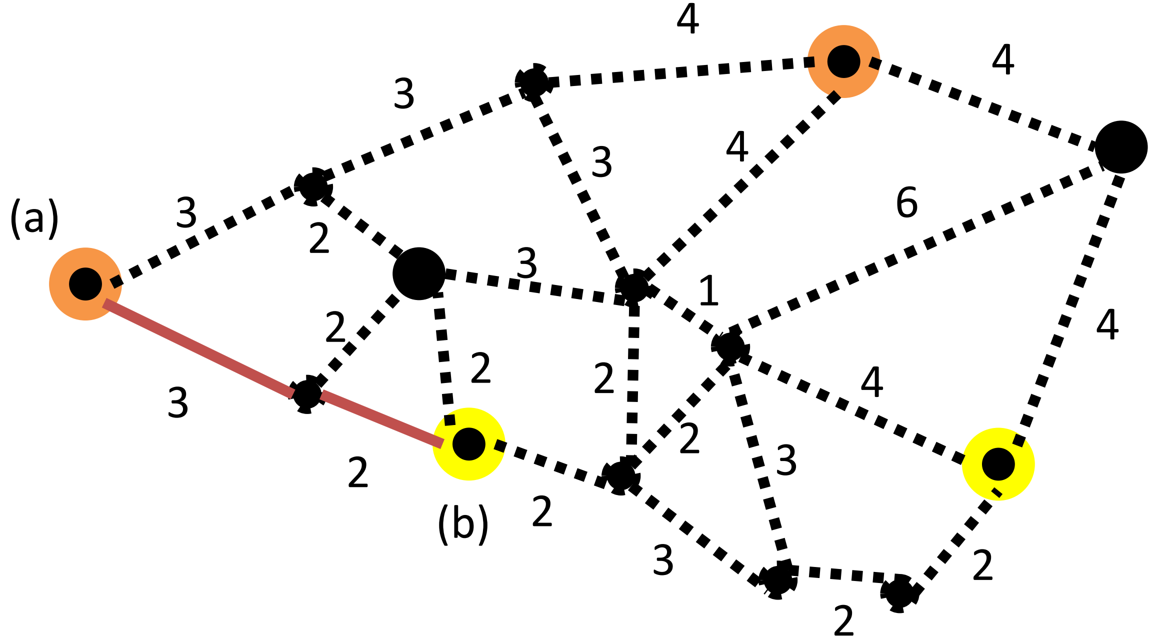
Approximation Algorithm II

select a terminal t not in T that is closest to a vertex in T



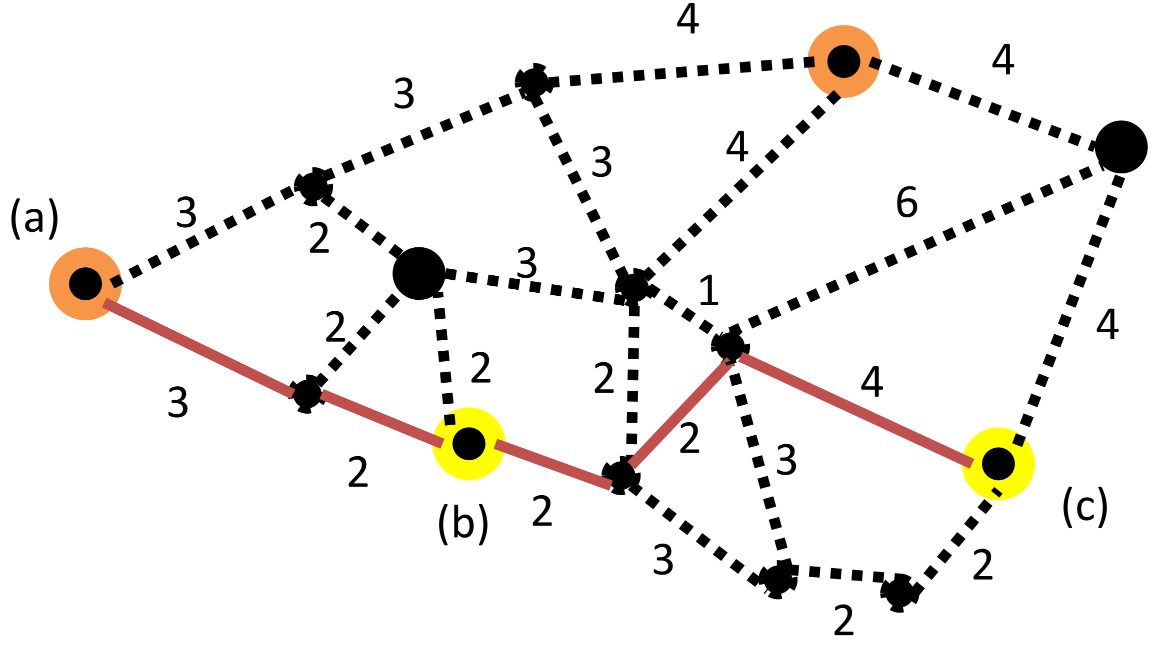
Approximation Algorithm II

select a terminal t not in T that is closest to a vertex in T



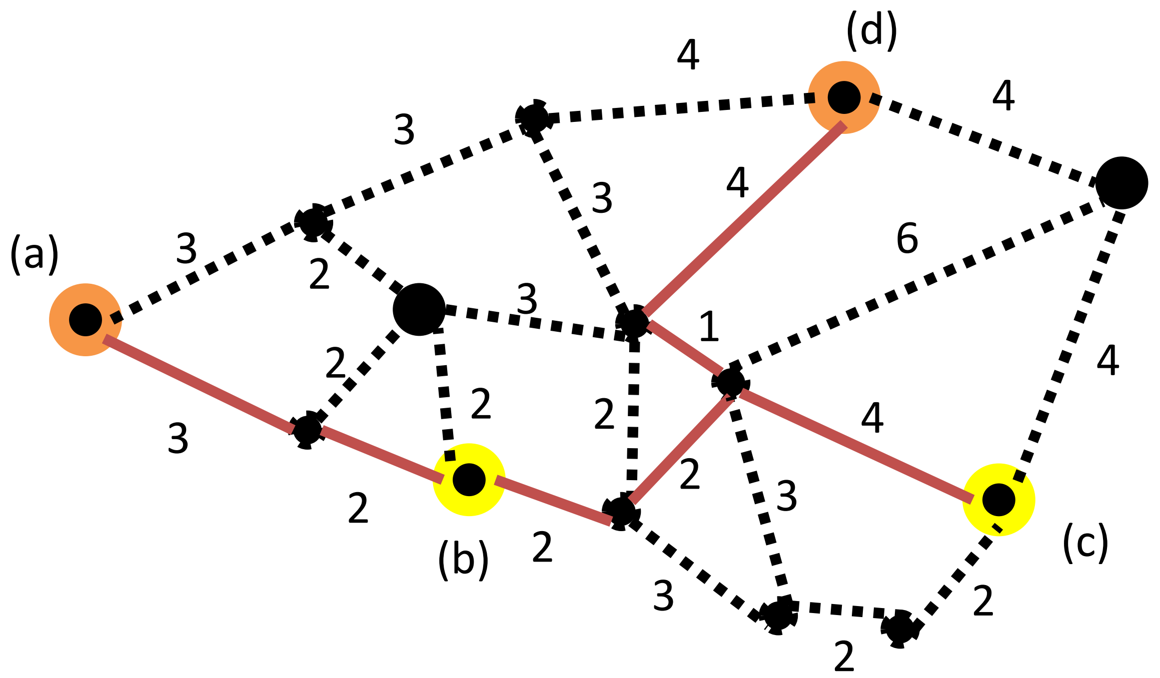
Approximation Algorithm II

select a terminal t not in T that is closest to a vertex in T



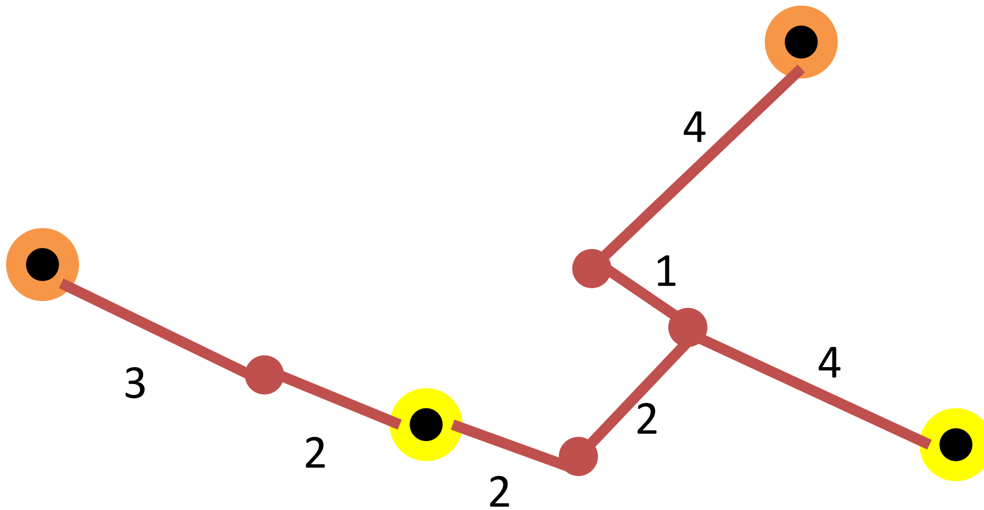
Approximation Algorithm II

select a terminal t not in T that is closest to a vertex in T



2-approximate Steiner Tree

build a subgraph of G=(V,E) induced by the vertices in found solution T



Cost= 5+9+4=18