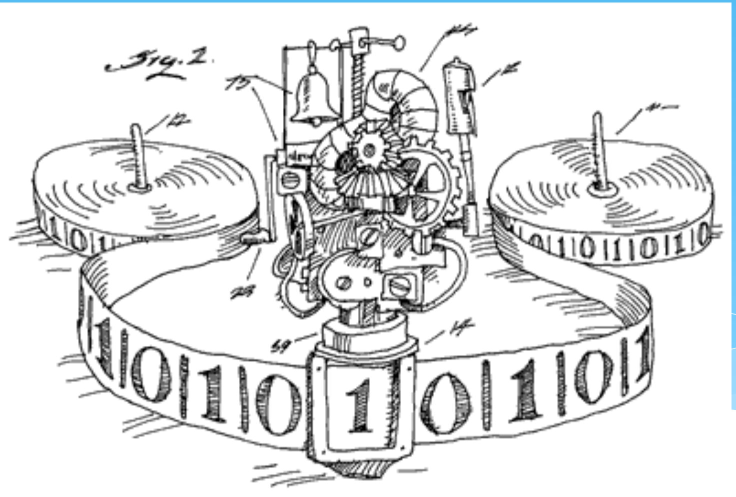
EECS 376: Foundations of Computer Science

Chris Peikert

23 January 2023



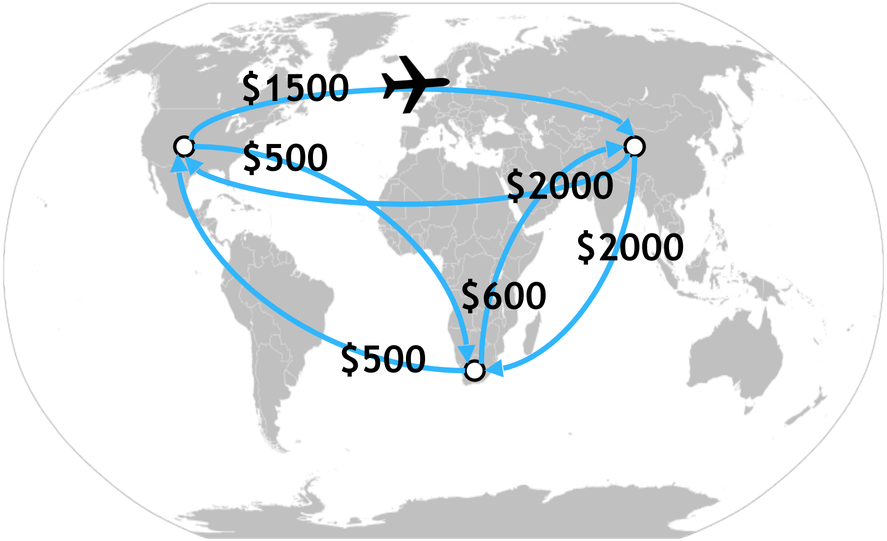
Today's Agenda

* Dynamic Programming for Shortest Paths
  * Floyd-Warshall All-Pairs SP
* Greedy Algorithms
  * Kruskal's Minimum Spanning Tree

The Flights Problem

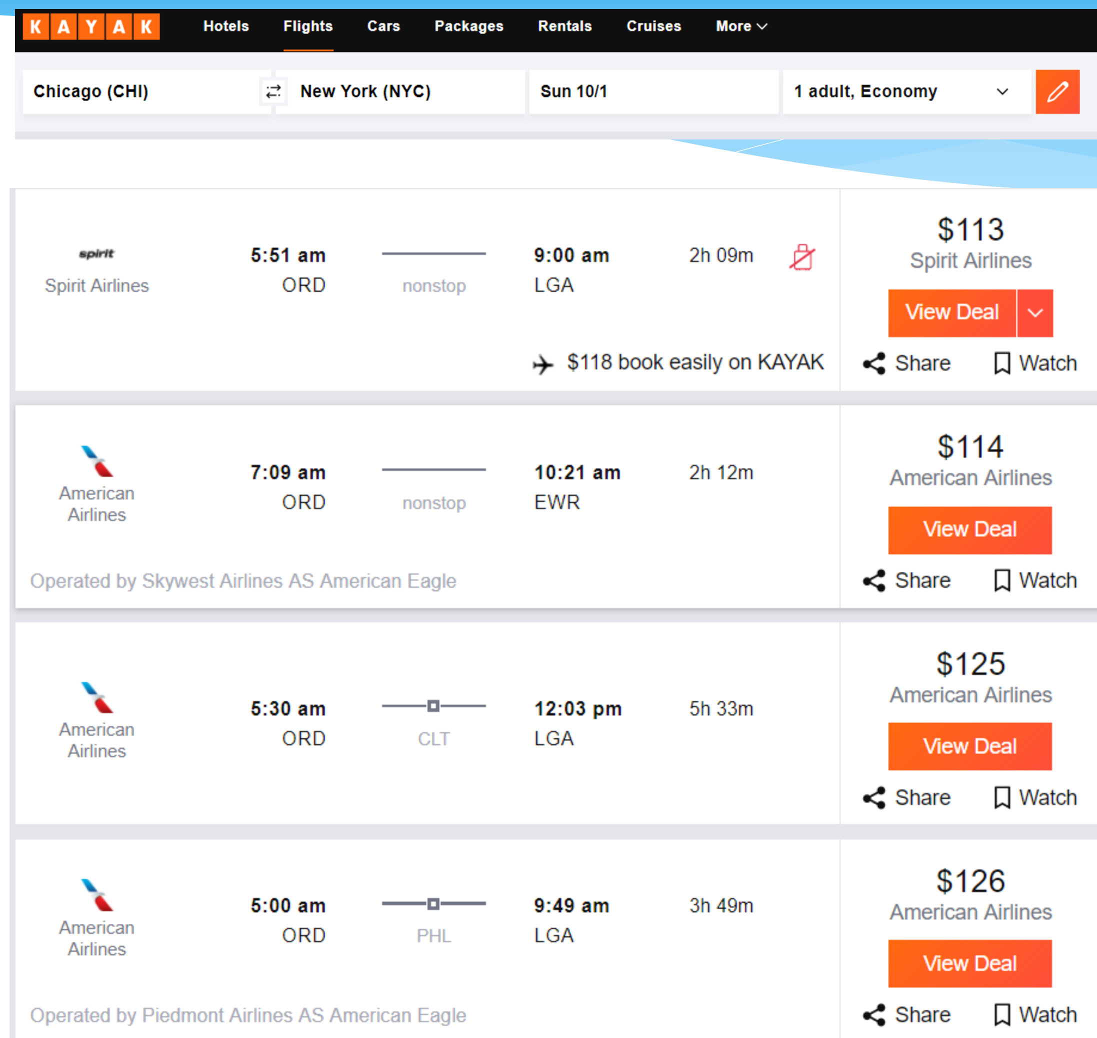
$c(i, j)$ might be different from $c(j, i)$

- Given: n airports and (possibly asymmetric) costs $c(i, j)$ to directly fly from airport i to j
- **Goal:** for *every* pair of airports i, j, find the *minimum* cost $d(i, j)$ to fly from airport i to j, *with "layovers" allowed*

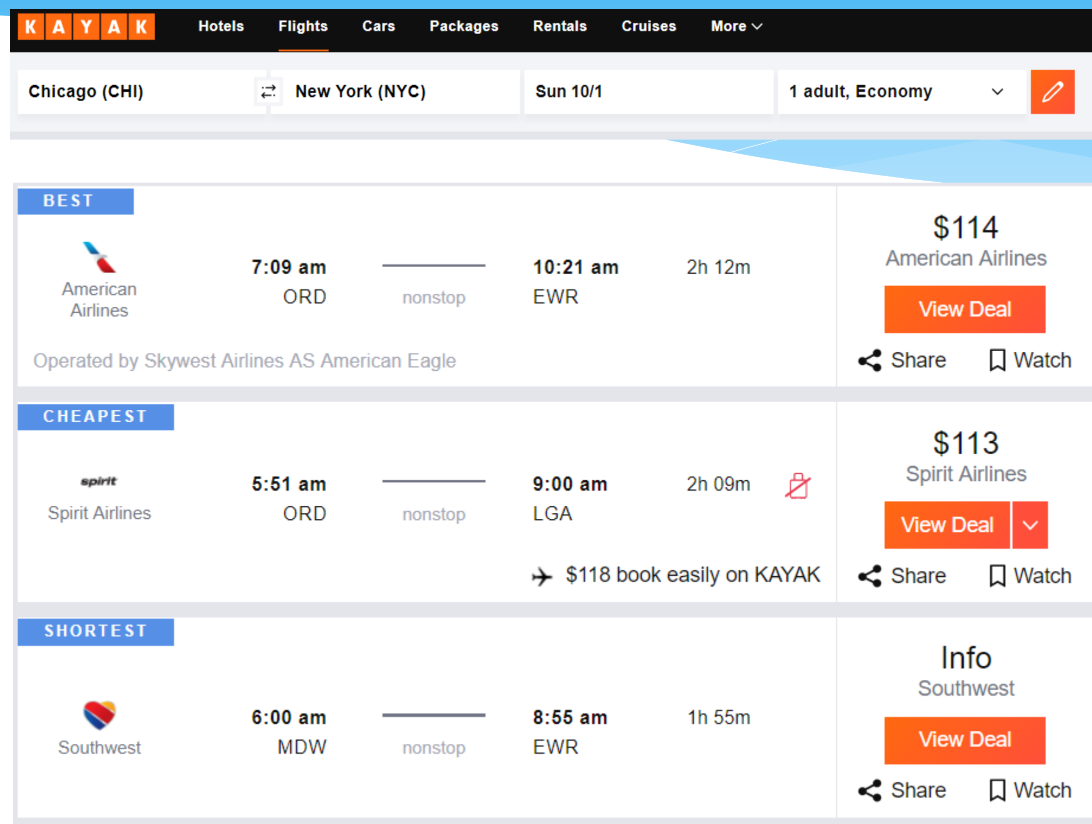


Example


Layover Airports
Atlanta (ATL)
Boston (BOS)
Buffalo (BUF)
Charlotte (CLT)
Cincinnati (CVG)
Cleveland (CLE)
Columbus (CMH)
Denver (DEN)
Detroit (DTW)
Fort Lauderdale (FLL)
Houston (IAH)
Las Vegas (LAS)
Los Angeles (LAX)
Minneapolis (MSP)
Myrtle Beach (MYR)
Nashville (BNA)
Norfolk (ORF)
Philadelphia (PHL)


KAYAK
Hotels
Flights
Cars
Packages
Rentals
Cruises
More
Chicago (CHI)
New York (NYC)
Sun 10/1
1 adult, Economy
spirit
Spirit Airlines
5:51 am
ORD
nonstop
9:00 am
LGA
2h 09m
$113
Spirit Airlines
View Deal
$118 book easily on KAYAK
Share
Watch
American Airlines
7:09 am
ORD
nonstop
10:21 am
EWR
2h 12m
$114
American Airlines
View Deal
Operated by Skywest Airlines AS American Eagle
Share
Watch
American Airlines
5:30 am
ORD
CLT
12:03 pm
LGA
5h 33m
$125
American Airlines
View Deal
Share
Watch
American Airlines
5:00 am
ORD
PHL
9:49 am
LGA
3h 49m
$126
American Airlines
View Deal
Operated by Piedmont Airlines AS American Eagle
Share
Watch

Example

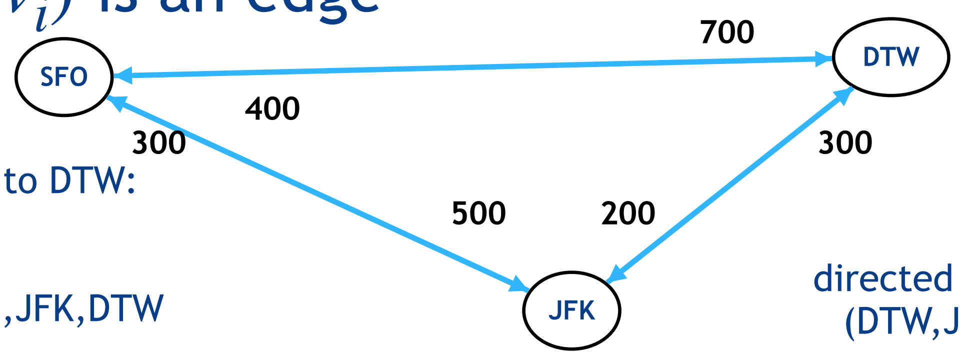


Layover Airports reset

- Atlanta (ATL)
- Boston (BOS)
- Buffalo (BUF)
- Charlotte (CLT)
- Cincinnati (CVG)
- Cleveland (CLE)
- Columbus (CMH)
- Denver (DEN)
- Detroit (DTW)
- Fort Lauderdale (FLL)
- Houston (IAH)
- Las Vegas (LAS)
- Los Angeles (LAX)
- Minneapolis (MSP)
- Myrtle Beach (MYR)
- Nashville (BNA)
- Norfolk (ORF)
- Philadelphia (PHL)

Review: Graph Theory

- A ***directed graph*** consists of some ***vertices*** and ***directed edges***: each from one vertex to another
- ***Weighted*** graph: each edge has a *weight*, or *cost* (in general, could be zero or negative)
- A ***path*** from vertex i to vertex j is a sequence of vertices $i = v_0, v_1, \ldots, v_\ell = j$ where each (v_{i-1}, v_i) is an edge

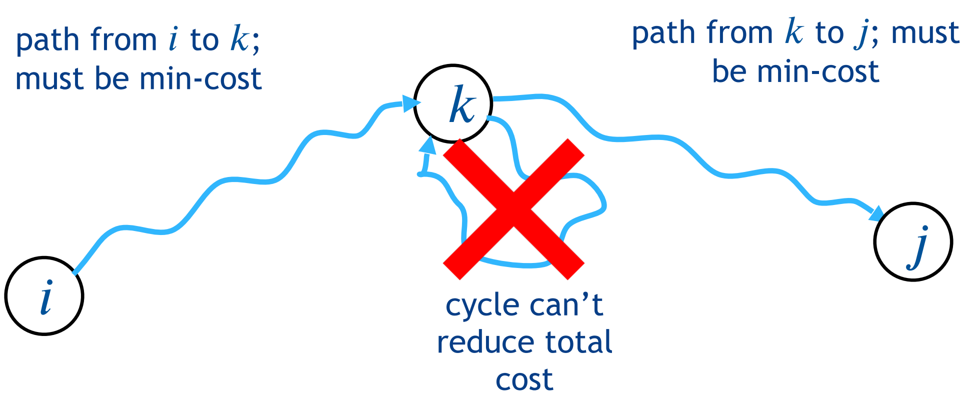


3 example paths SFO to DTW:
- SFO,DTW
- SFO,JFK,DTW
- SFO,DTW,JFK,SFO,JFK,DTW

directed edge (DTW,JFK)

Two Key Observations

- Consider some cheapest path from i to j.
  - **1.** If it goes through k, then it must take a cheapest path from i to k, then a cheapest path from k to j.
  - **2.** If the graph has *no negative-cost cycles*, then (wlog) there are *no duplicate vertices* in the path.



Shortest-Path Subproblems

Consider a min-cost path from i to j.
1. If it goes through k, then it takes a min-cost path from i to k, and a min-cost path from k to j.
2. If the graph has no *negative-cost cycles*, then (wlog) there are *no duplicate vertices* in the path.

- Given: n vertices (airports) and edge (flight) costs $c(i, j)$
- **Definition:** Let $d^k(i, j)$ be the minimum cost from i to j allowing only vertices $1, \ldots, k$ in between
  - **Example:** For $k = 0$, only direct flights allowed (no layovers).
  - **Example:** For $k = 3$, layovers allowed only in ATL, BOS, and BUF.

Shortest-Path Recurrence

Consider a min-cost path from i to j.
1. If it goes through k, then it takes a min-cost path from i to k, and a min-cost path from k to j.
2. If the graph has no *negative-cost cycles*, then (wlog) there are *no duplicate vertices* in the path.

- **Base Case (Direct Hops):** $d^0(i,j) = c(i,j)$ for all i, j
- **Recursive Case, over k:**
  - Consider a cheapest path from i to j with only $1,\dots,k$ allowed in between.
  - Only two possibilities: 1. it either uses k (once) in between, or 2. it doesn't.
  - **1.** It uses a cheapest path from i to k, then a cheapest path from k to j: $d^k(i,j) = d^{k-1}(i,k) + d^{k-1}(k,j)$.
  - **2.** It's a cheapest path from i to j using only $1,\dots,k-1$ in between: $d^k(i,j) = d^{k-1}(i,j)$
  - Which possibility is it? Cheapest one!

Floyd–Warshall (1962) Algorithm

Recurrence:

$$d^k(i,j) = \begin{cases} c(i,j) & \text{if } k = 0 \\ \min\{d^{k-1}(i,j), d^{k-1}(i,k) + d^{k-1}(k,j)\} & \text{if } 0 < k \leq n \end{cases}$$

Bottom-up Algorithm:

Floyd-Warshall($c[1\ldots n][1\ldots n]$):
allocate $D[1\ldots n][1\ldots n][0\ldots n]$
for all i, j: $D[i][j][0] = c[i][j]$
for $k = 1\ldots n$ and all i, j:

$$D[i][j][k] \leftarrow \min\left\{\begin{matrix} D[i][k][k-1] + D[k][j][k-1], \\ D[i][j][k-1] \end{matrix}\right\}$$

Runtime:
$O(n^3)$
(better?)

https://www.cs.usfca.edu/~galles/visualization/Floyd.html

“Greed, in all of its forms -- greed for life, for money, for love, knowledge -- has marked the upward surge of mankind..”
- Gordon Gekko, *Wall Street (1987)*

Algorithmic Strategy:
Be Greedy

Warning! Greed Never Pays

(except when it does)

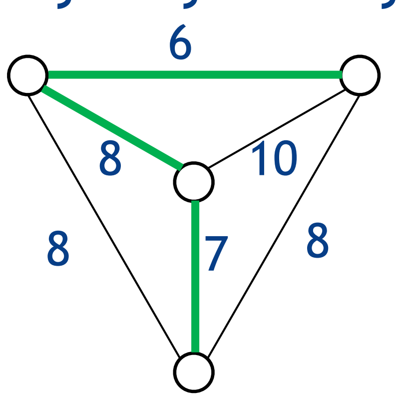
Greedy Template

- Solve a problem in a “greedy” / “optimistic” way.
  - Hope that making “locally optimal” decisions ultimately yields a global optimum.
  - Rarely yields a correct algorithm, but can be very elegant when it does.
- **Main Difficulty:** arguing correctness
  - Exchange arguments

A Connection Problem

$d(i, j) = d(j, i)$

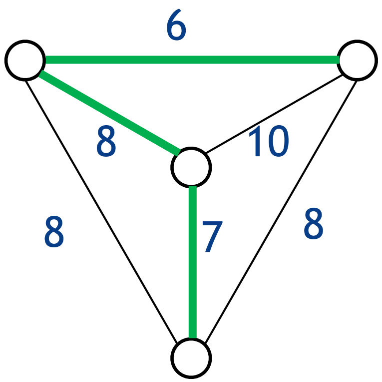
- Given n cities with *symmetric*, positive distances (costs) $d(i, j)$ between cities i and j.
- **Goal:** Find the *minimum* length of highway needed to ***connect*** all the cities—i.e., possible to drive from any city to any another using the highways



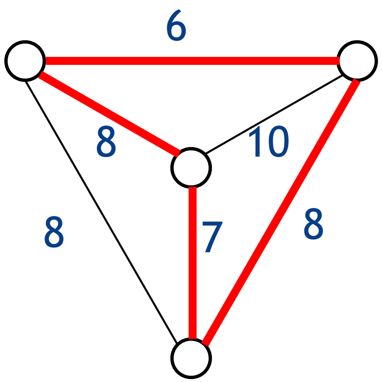
Q: What's the minimum length of highway needed here?

Review: Graph Theory

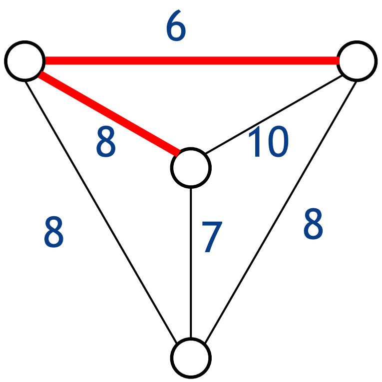
- **Fact:** Every ***connected*** undirected graph G has a ***spanning tree***: a *connected* subgraph that (i) has *every vertex* of G, and (ii) has *no cycles*.
- **Goal:** Find a *minimum-weight* spanning tree (***MST***)



(i) and (ii)



(i) but not (ii)



(ii) but not (i)

Kruskal's Algorithm

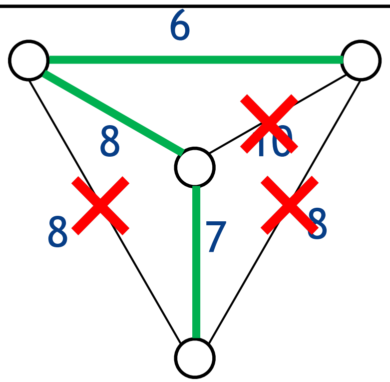
Kruskal(G)**:** // G is a weighted, undirected graph

$T \leftarrow \varnothing$ // invariant: T has no cycles

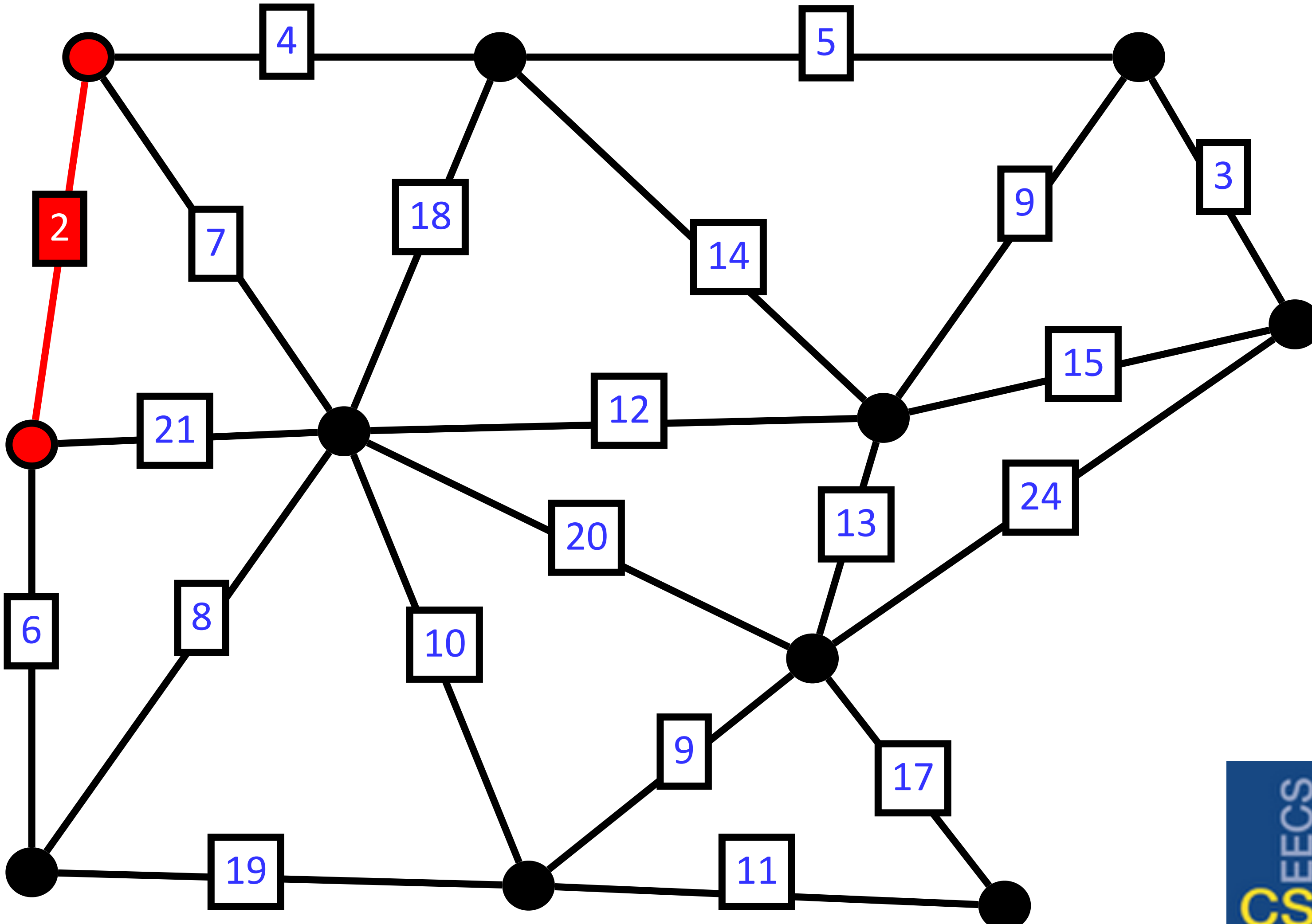
for each edge e in *non-decreasing order by weight*:

    if $T + e$ is acyclic: $T \leftarrow T + e$

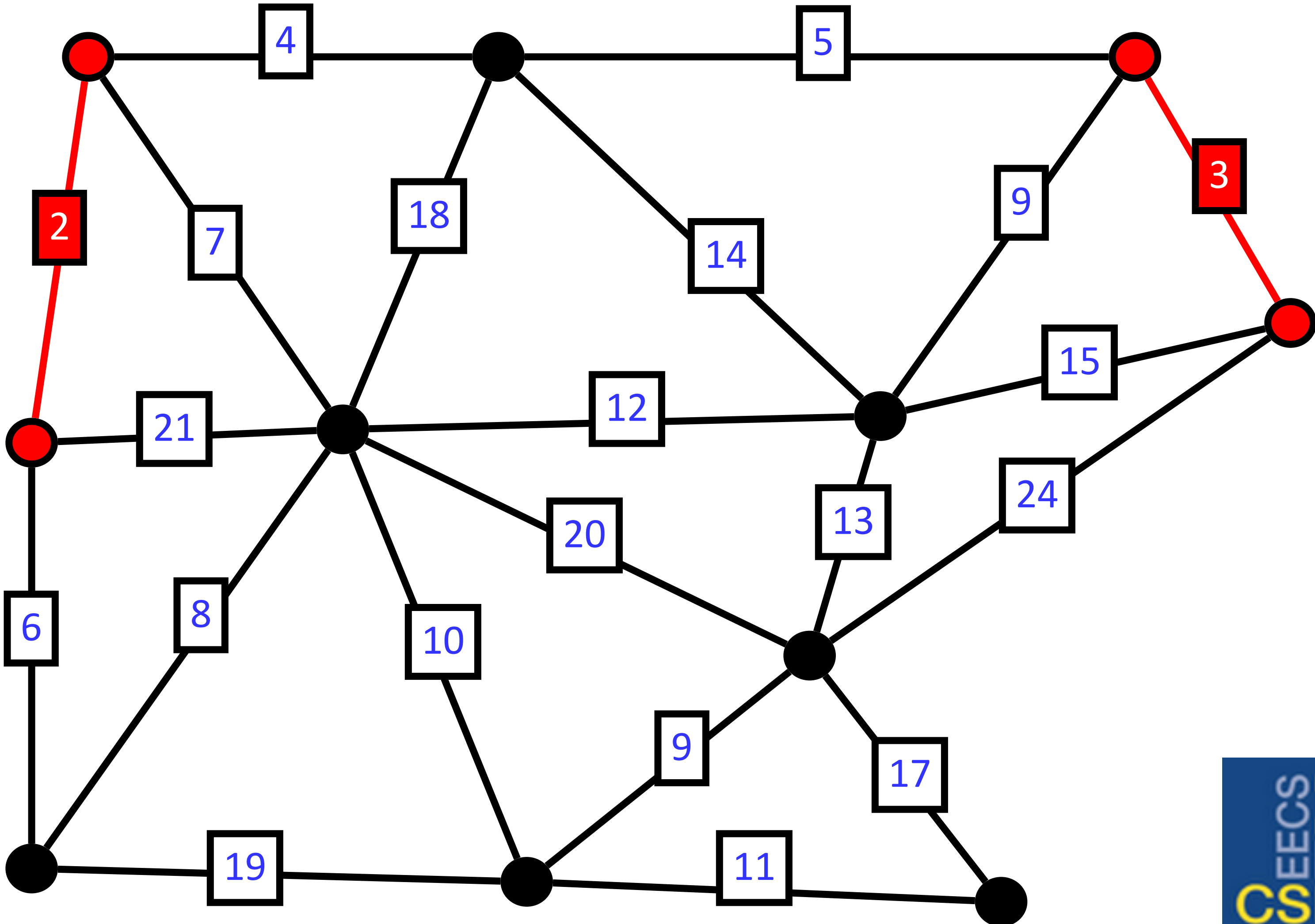
return T



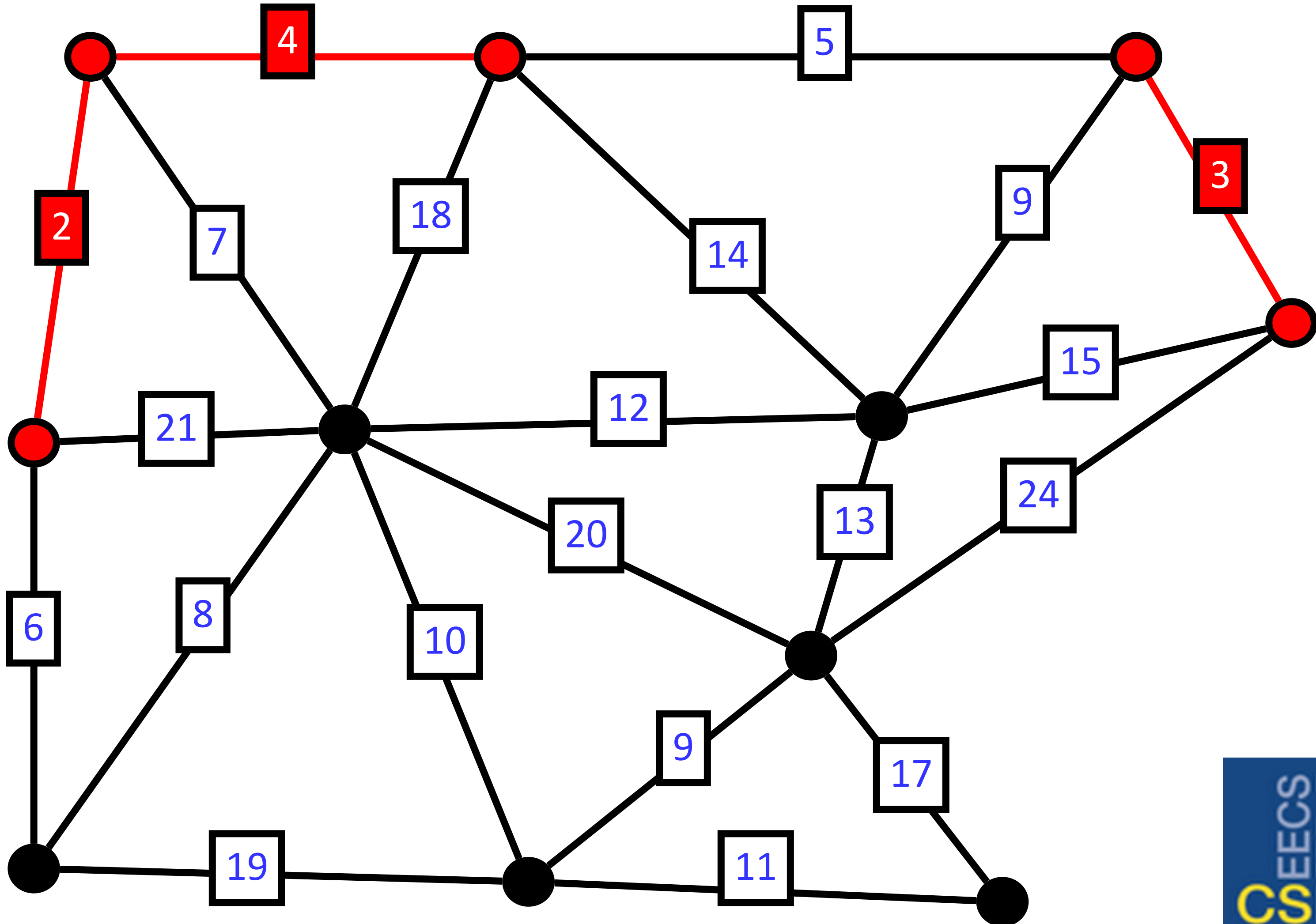
Kruskal's Algorithm



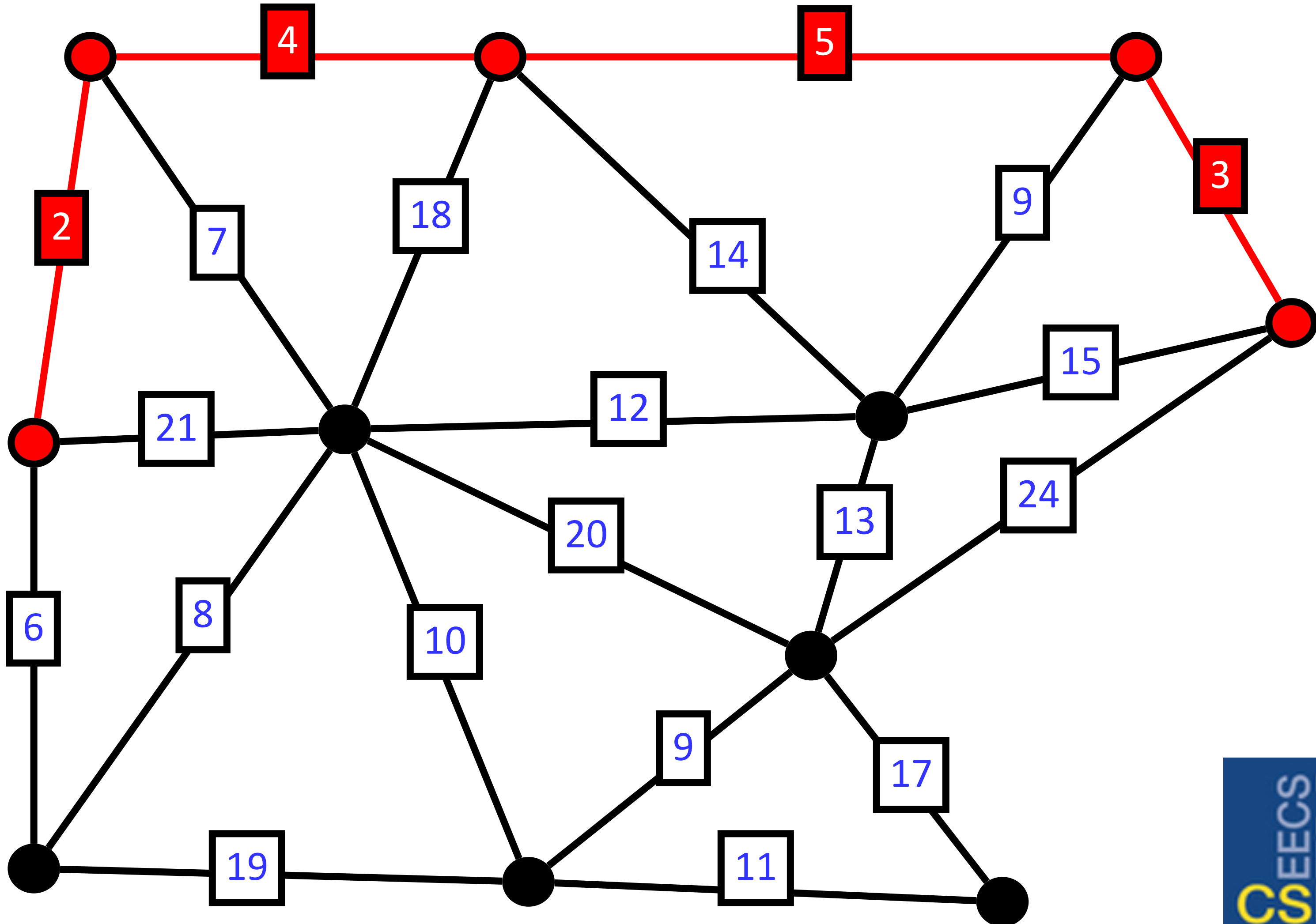
Kruskal's Algorithm



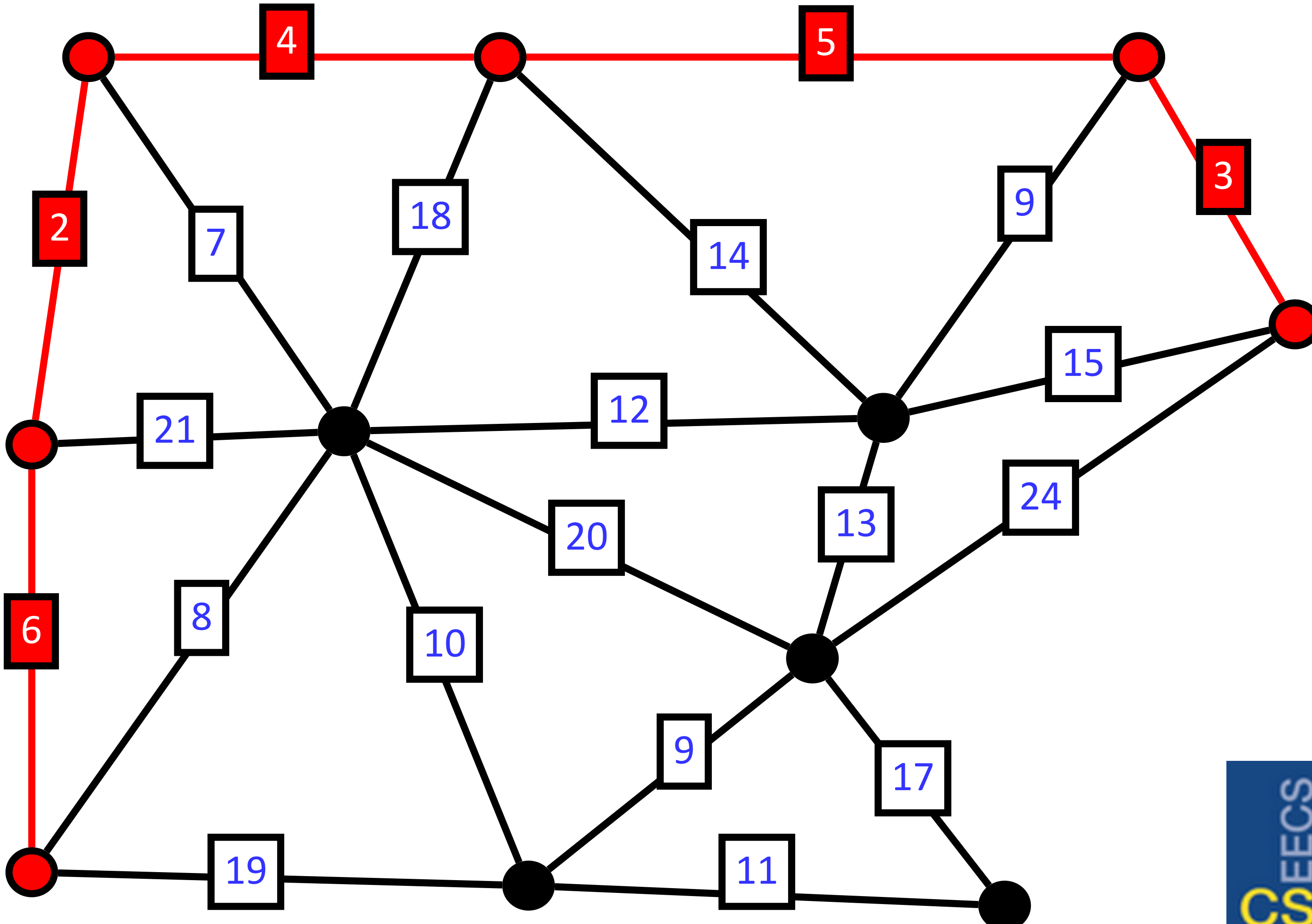
Kruskal's Algorithm



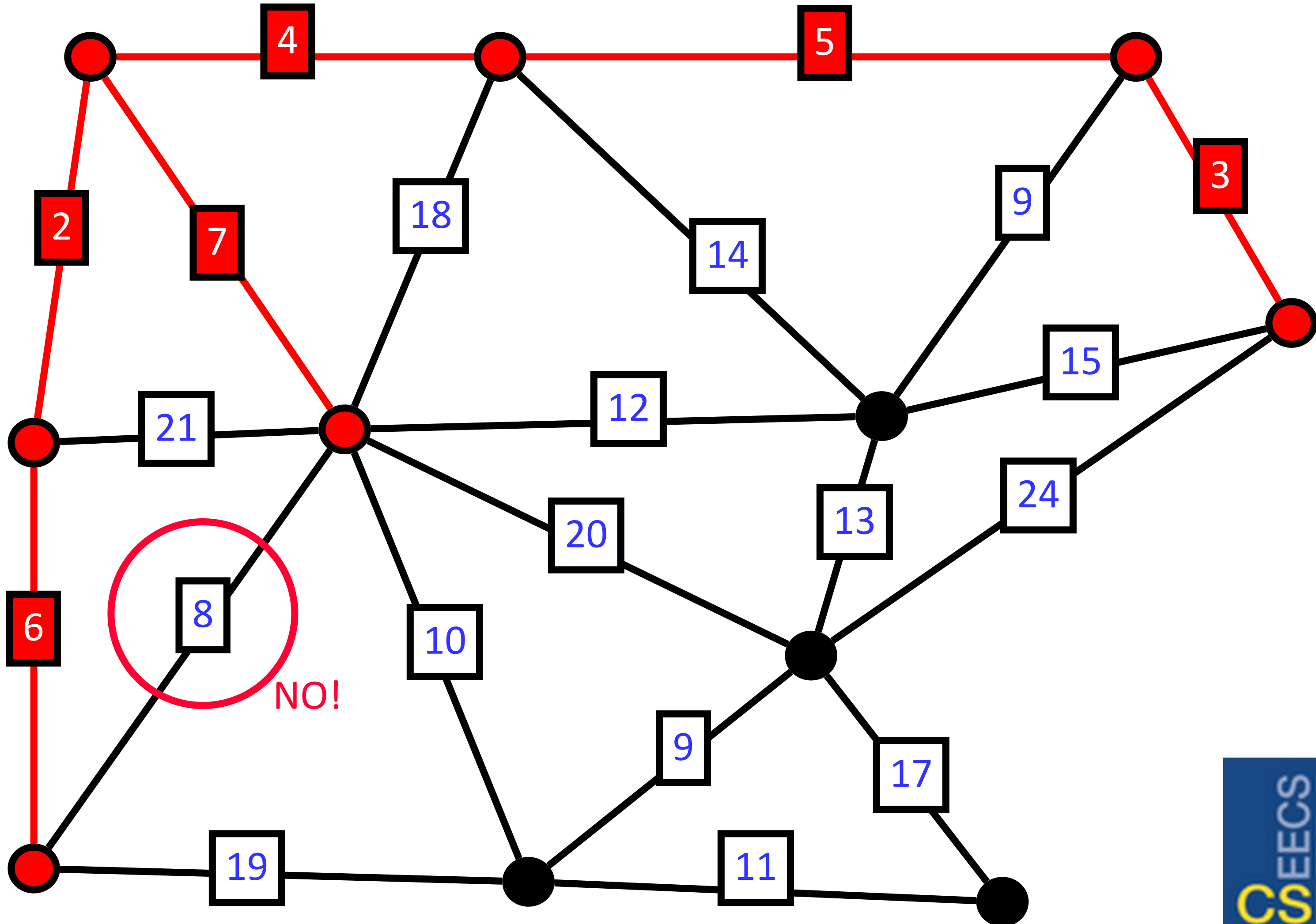
Kruskal's Algorithm



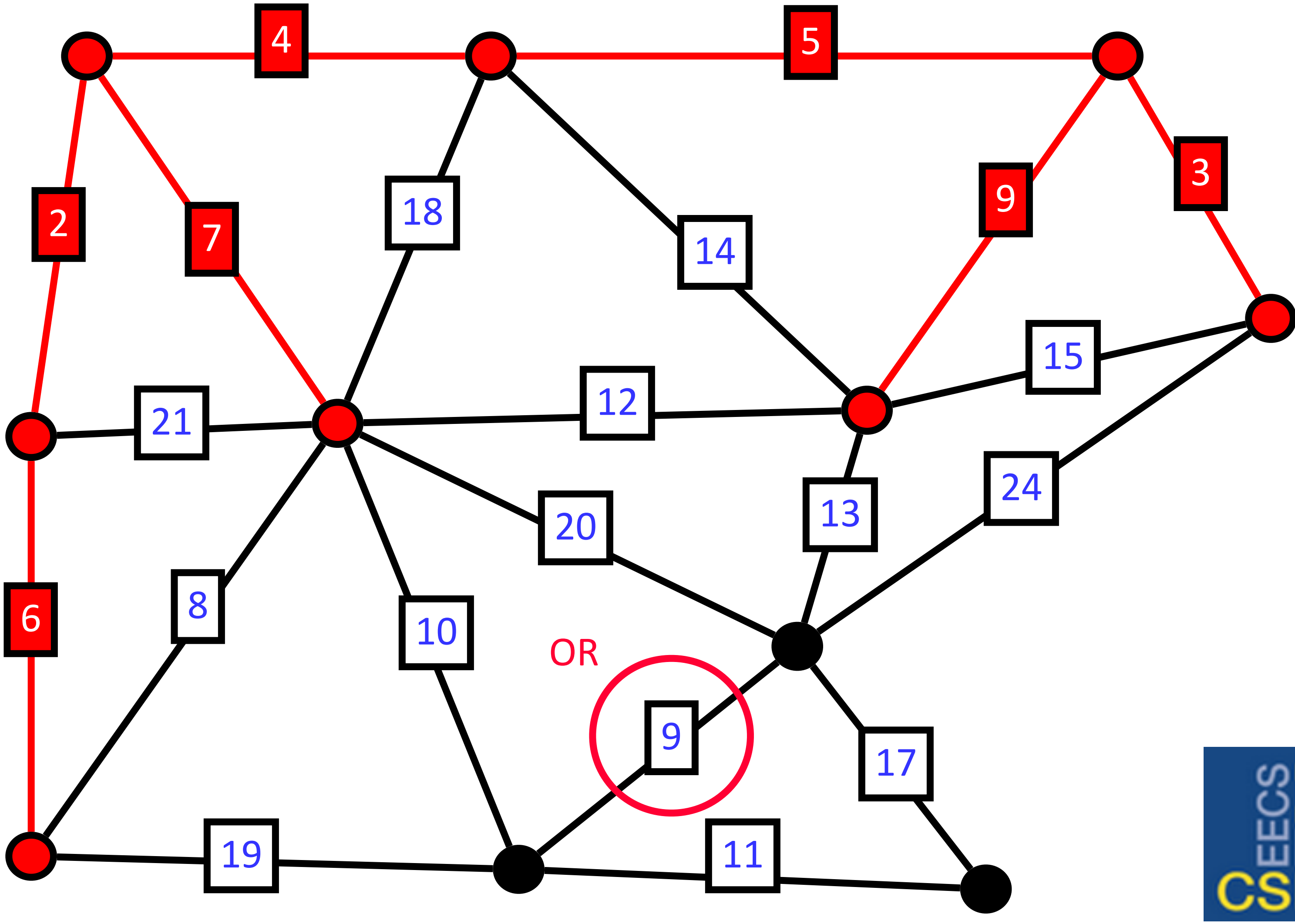
Kruskal's Algorithm



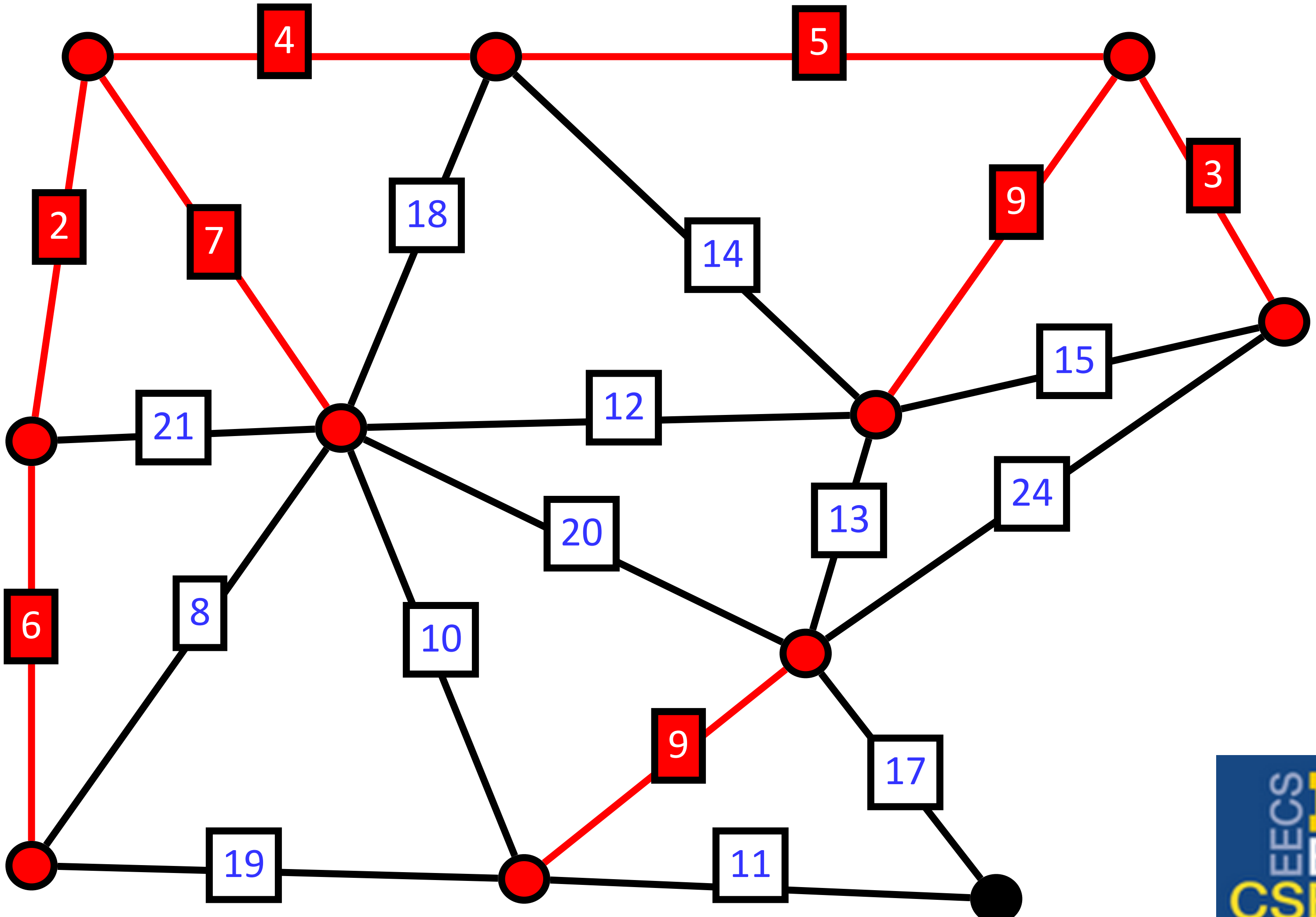
Kruskal's Algorithm



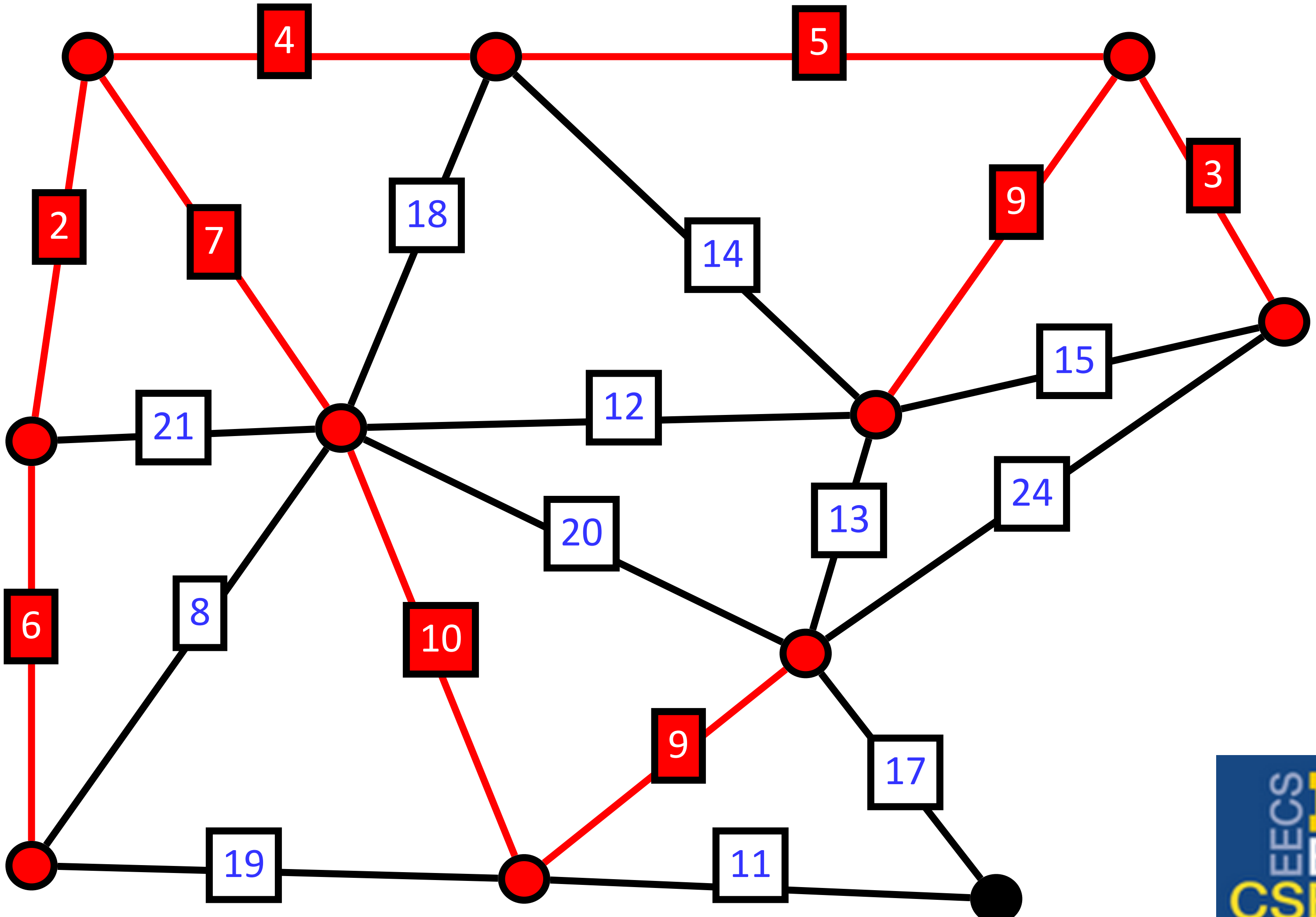
Kruskal's Algorithm



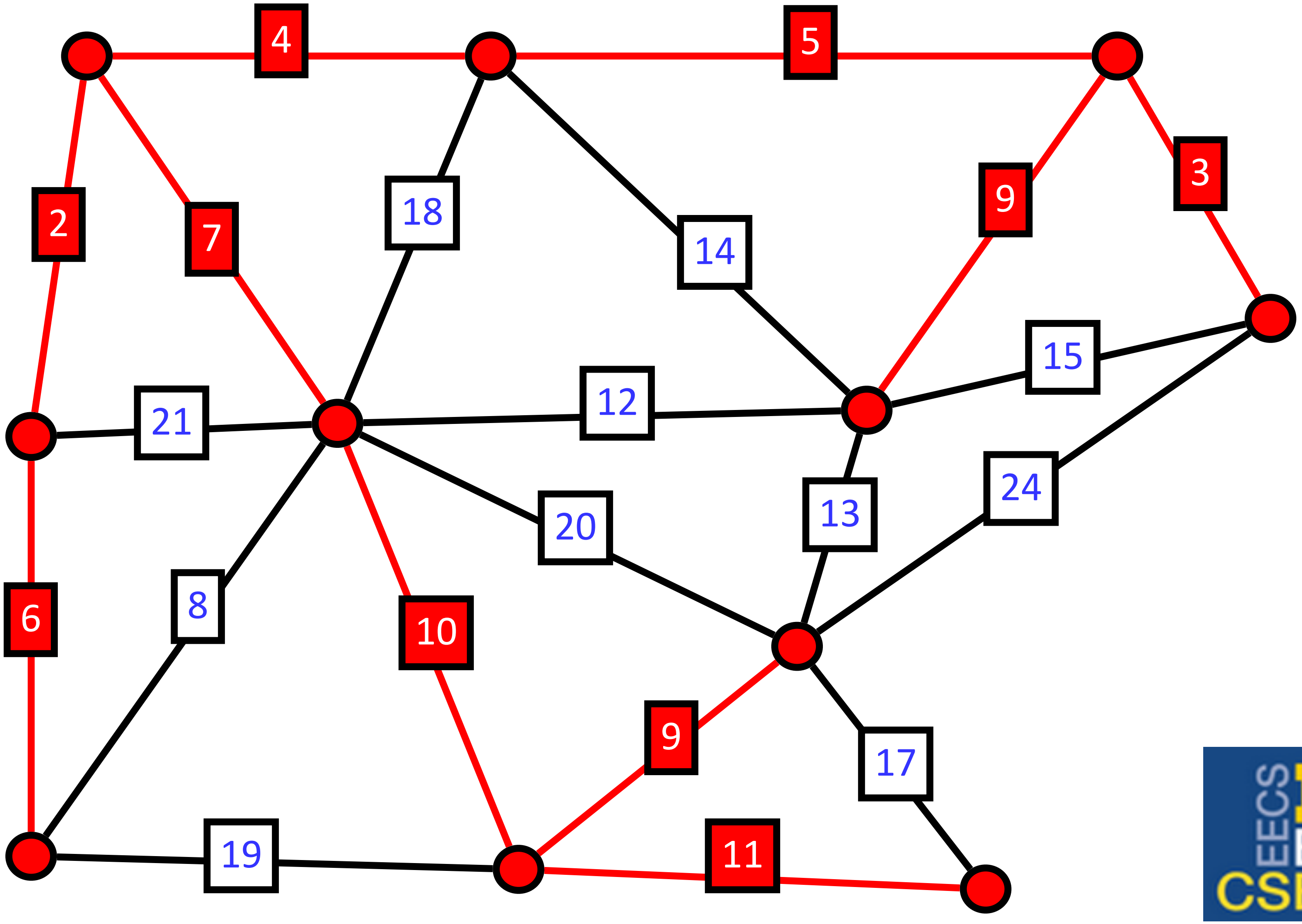
Kruskal's Algorithm



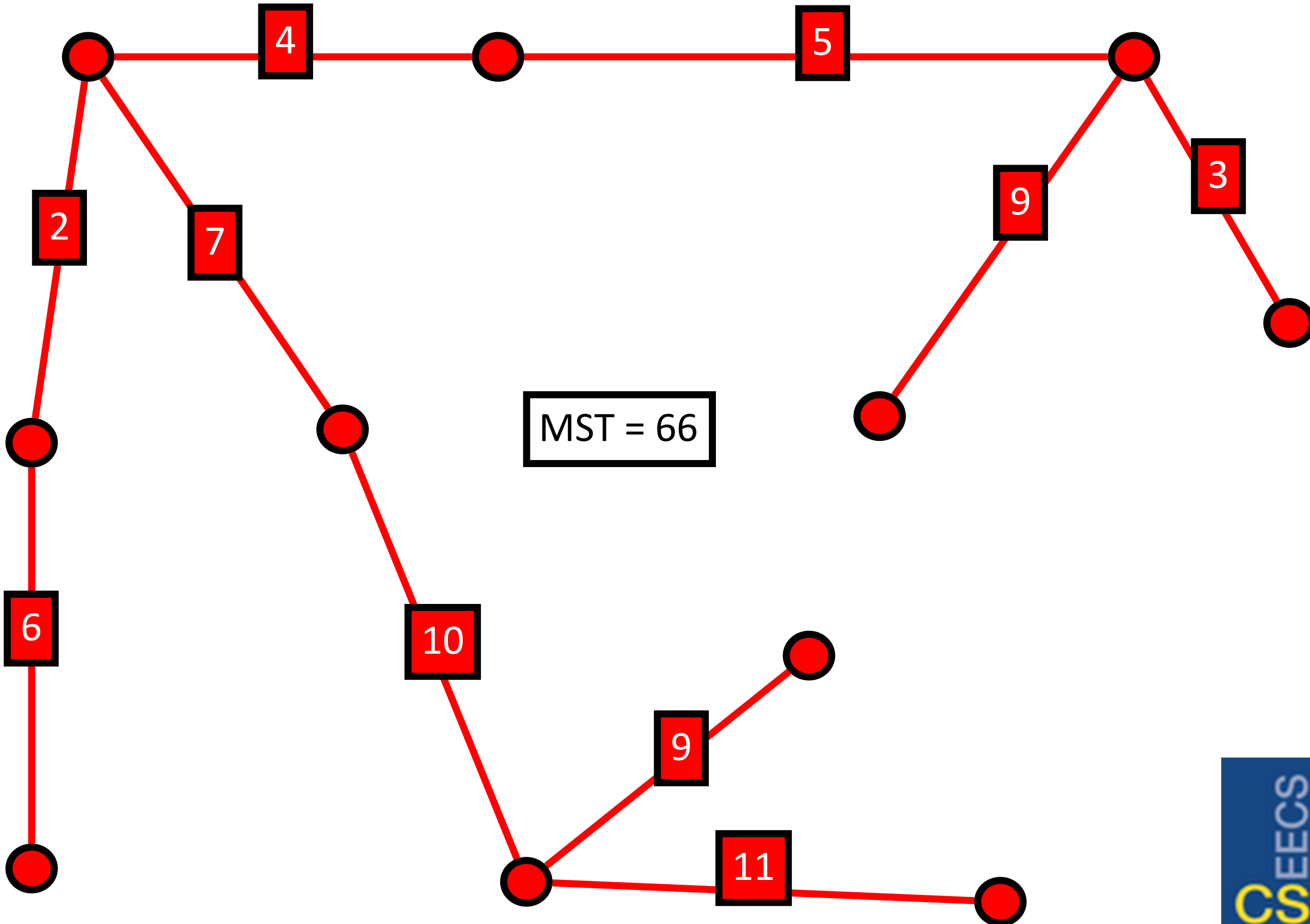
Kruskal's Algorithm



Kruskal's Algorithm



Kruskal's Algorithm



Kruskal's Algorithm: Correctness

Kruskal(G): // G is a weighted, undirected graph
$T \leftarrow \varnothing$ // invariant: T has no cycles
for each edge e in *increasing order of weight*:
  if $T + e$ is acyclic: $T \leftarrow T + e$
return T

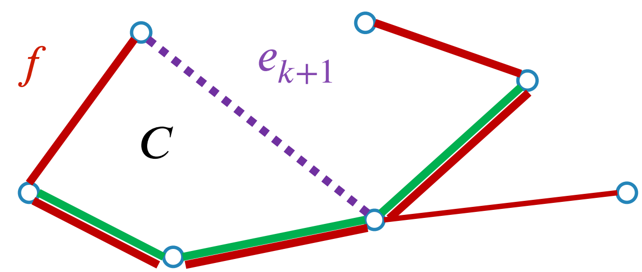
- **Exercise:** **Kruskal**(G) returns a *spanning tree* T of G
- Suppose T' is an *arbitrary* MST of G.
- **Goal:** Show weight of $T =$ weight of T'.
- **Idea:** Show we can *transform* T' to T, *maintaining an MST* (by induction, "swapping in" an edge of T for one of T', one at a time)
- This is a commonly employed strategy to show that a greedy algorithm is optimal.

Kruskal returns an MST

Kruskal(G): // G is a weighted, undirected graph
$T \leftarrow \varnothing$ // invariant: T has no cycles
for each edge e in *increasing order of weight*:
  if $T + e$ is acyclic: $T \leftarrow T + e$
return T

- Let $e_1, e_2, \ldots$ be the edges of T, *in order of addition to* T (so, non-decreasing by weight).
- Let $f_1, f_2, \ldots$ be the edges of T', *in non-decreasing order by weight*.
- **Base case:** First $k = 0$ edges are trivially the same.
- **Inductive step:** Suppose first k edges are the same: $e_i = f_i$ for $i = 1, \ldots, k$.
  - Consider the next edge e_{k+1} added to T. If $e_{k+1} \in T'$, then wlog $e_{k+1} = f_{k+1}$ (why?).
  - If $e_{k+1} \notin T'$, then adding it to T' creates a cycle C.
  - Since T is acyclic, on the cycle C there is an edge $f \in T', f \notin T$.
  - "Swap in e_{k+1}": Remove f from T', include e_{k+1}. Now can make $f_{k+1} = e_{k+1}$ (how?).

Claim: e_{k+1}'s weight $\leq f$'s weight.
Proof: e_{k+1} is the next edge to be added to T, and we add edges *in non-decreasing order of weight*!



The Coin Change Problem

- An exotic country uses \$1, \$5, \$10, \$25 coins.
- **Goal:** Make change for $n \geq 1$ dollars using these denominations, using fewest number of coins possible.
- **Greedy:** repeatedly give largest coin possible, until done.
- **Correctness:** Must prove that *no strategy can do better*.
- **Idea:** Show that an arbitrary optimal solution can be transformed to the greedy solution without increasing the number of coins (**next HW?**).
- **Warning:** There are some coin denominations for which greedy is not optimal!

Goodbye Algorithms…

- Congratulations! You've just finished a crash course on algorithms. (Practice, practice, practice!)
- If you'd like to know/do more, consider EECS 477, *Introduction to Algorithms*.
- Some open problems:
  - Longest Common Subsequence in $o(mn)$?
  - All-pairs shortest paths in $O(n^{2.9999})$?
  - Minimum Spanning Tree in $O(m)$?
    - Pettie-Ramachandran [2002]: optimal—but unknown!—running time!