We have been considering NP-hard problems recently such as vertex cover (VC), the traveling salesman problem (TSP), and independent set (IS). Most researchers believe that it is not possible to design polynomial time algorithms for these problems.

The best known algorithms are generally too slow to be of use in practice unless the input size is very small. What can we do with these problems?

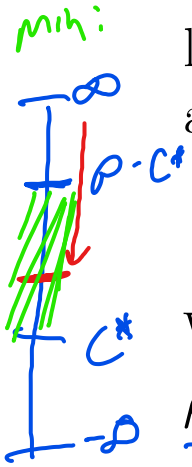
One approach is to develop a *heuristic* which runs very fast, but has no guarantee on the quality of the solution (will typically return good results, but could be really bad).

Another approach is to develop polynomial time algorithms which guarantee that the solution computed by the algorithm will not be too much worse than an optimal solution. Such an algorithm is called an **approximation algorithm**.

Consider the optimization variants of VC, TSP, and IS:

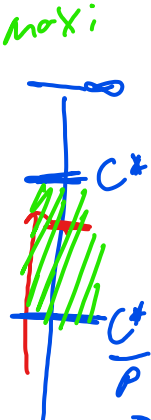
- Compute a smallest vertex cover.
- Compute a shortest tour.
- Compute a largest independent set.

Let C^* denote the value of an optimal solution, and let C denote the value of a solution returned by an approximation algorithm.



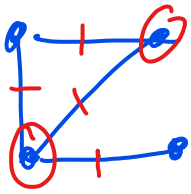
We say that the algorithm has *approximation ratio* $\rho > 1$ if we can guarantee

- $C \leq \rho \cdot C^*$ for minimization problems, or
- $C \geq C^*/\rho$ for maximization problems.



In either case, the guarantee is that the returned solution will be within a factor of ρ. We would like to develop polynomial time approximation algorithms with an approximation ratio as small as possible.

We will now focus on the optimization version of the VC problem (we want to find a vertex cover of minimum cardinality).



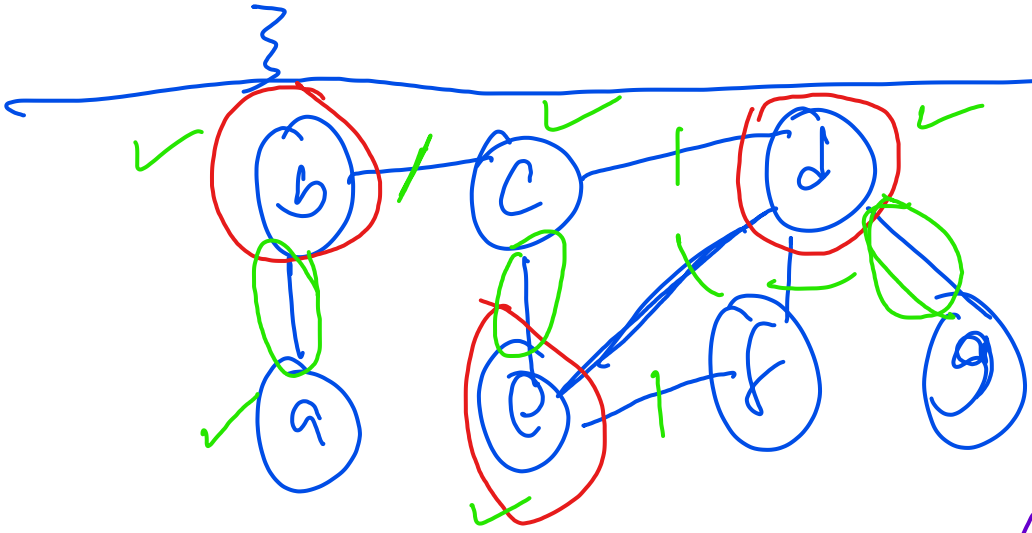
Again, let C^* denote the value of an optimal solution, and let C denote the value of a solution returned by our algorithm.

We will give an approximation algorithm with approximation ratio 2. That is, we will give an algorithm such that $C \leq 2 \cdot C^*$.

For short, we call such an algorithm a 2-approximation (an approximation algorithm with approximation ratio ρ is called an ρ-approximation in general).

Consider the following algorithm:

```
VC-Approx(G=(V,E))
{
    C = ∅        // A = ∅
    E' = E
    while(E' ≠ ∅)
    {
        Let {u,v} be any edge of E'.   Add this edge to A.
        Add both u and v to C.
        Remove from E' all edges incident to either u or v.
    }
    return C
}
```



$C^* = 3$

$C = 6$

A is the circled edges.

$|A| = 3$

Claim: $|A| \leq C^*$

Theorem: VC-approx is a polynomial-time 2-approximation algorithm.

Running time: $O(n+m)$

Output is a VC because E' is empty when we terminate, and we only removed an edge from E' when we chose one of its endpoints.

It must be that $|A| \leq C^*$, because each edge in A must be covered by one of its endpoints, but a vertex is an endpoint for at most one edge of A. Therefore we need at least $|A|$ vertices to cover the edges of A.

Our solution has size $2|A|$.

$\Rightarrow$ Our solution is of size $\leq 2 \cdot C^*$

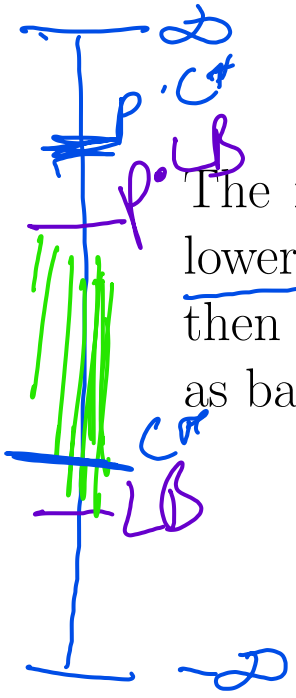
(1+ε) 1.01 ⇒ ε = 0.01 $n^{\frac{1}{\varepsilon}}$ or $n^{(\frac{1}{\varepsilon})^2}$

Now consider TSP with the triangle inequality.

- Triangle inequality: $c(u, v) \leq c(u, w) + c(w, v)$

The triangle inequality holds for many applications, including if the cost of moving from u to v is the Euclidean distance between the points in the plane.

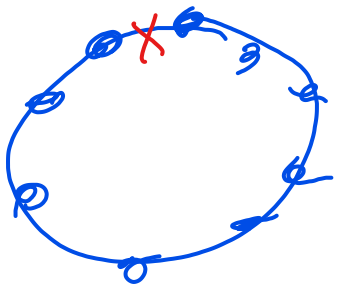
We will give a 2-approximation algorithm for this problem.



The idea is similar to last time. We will compute a lower bound on the cost of an optimal tour, and we will then compute a solution whose cost is at worst 2 times as bad as the lower bound.

OPT TSP ≥ Spanning tree obtained by deleting any edge ≥ MST

If we delete any edge from an optimal tour, we are left with a spanning tree which costs less than the tour. Therefore the minimum spanning tree must cost less that the minimum TSP tour.



Consider the following algorithm:

TSP-Approx(G=(V,E))

{ Select a root vertex r.

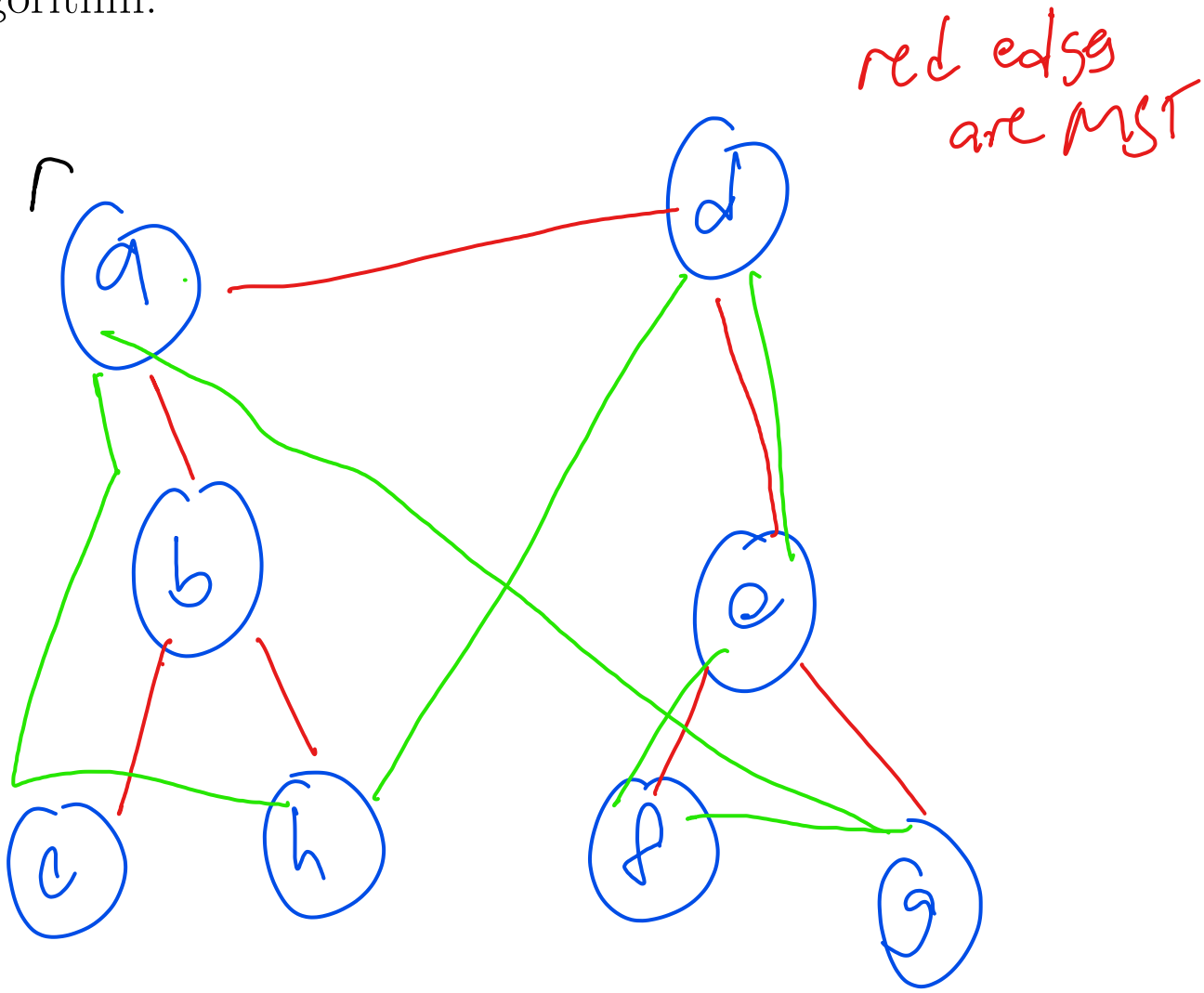
Compute an MST T for G from r using Prim's Algorithm.

Let H be a list of vertices ordered according to when they are first visited in a preorder tree walk of T.

return H

}

Theorem: TSP-approx is a polynomial-time 2-approximation algorithm.



a b c b h b a d e f e g e d a

Tour: a, b, c, h, d, e, f, g

.