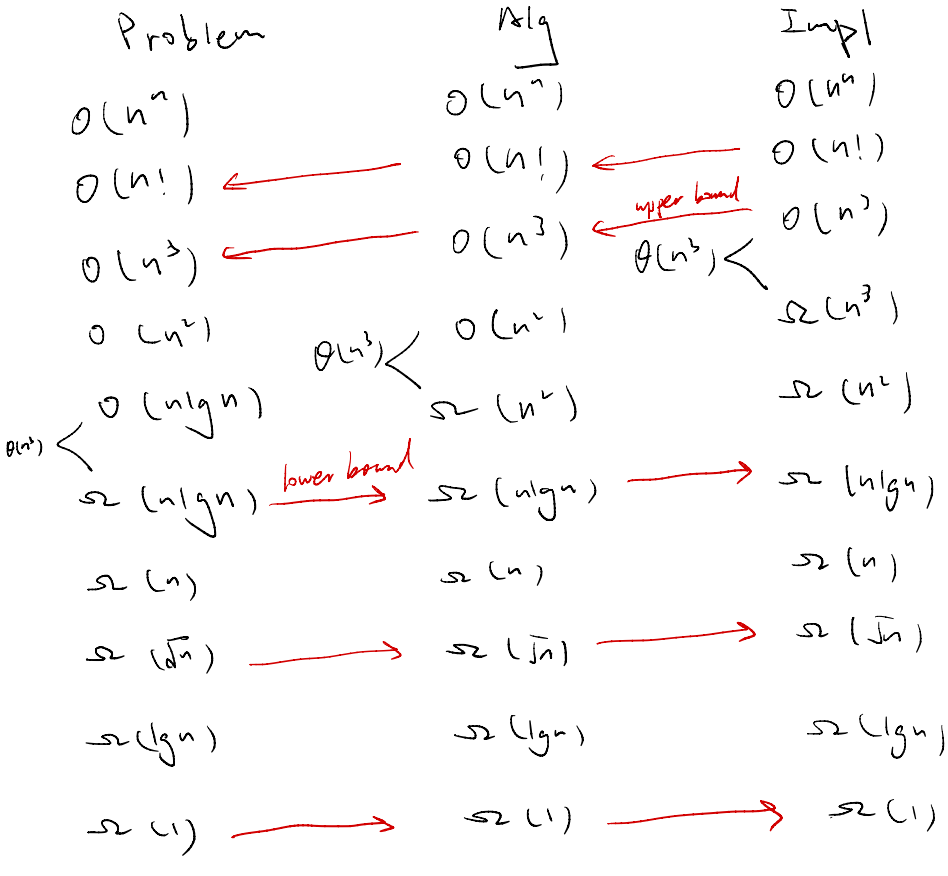
Lecture_3 02.08.23



$11 \% ? = 4$

$-1 \% 5 = 4$

$9\,7\,2\,6\,3\,4 \% 9 = 4$

$\times \quad 4\,1\,3\,5\,7\,2 \% 9 = 4$

$\% 9 = (4 \times 4) \% 9 = 16 \% 9 = 7$

$9\,7\,2\,6\,3\,\underline{4} \% 5 = 4$

$\times \quad 4\,1\,3\,5\,\underline{7\,2} \% 5 = 2$

$\% 5 = 2 \times 4 \% 5 = 8 \% 5 = 3$

$2^1 \% 11 = 2 \qquad 2^4 \% 11 = 5$

$2^5 \% 11 = 2 \cdot 2^4 \% 11 = 2^1 \% 11 \times 5 = 2 \times 5 = 10$

Lecture_3

1. ~~Finding the median requires sorting the array~~ and then using the $\frac{(n+1)}{2}th$ element or mean of n & $\frac{(n+1)}{2}$ th element. Sorting an array can solve median therefore it is just as hard or harder.

   ❌

   You can find the median by sorting the array and then using the $\frac{(n+1)}{2}th$ element or mean of n & $\frac{(n+1)}{2}$ th element. Sorting an array can solve median therefore it is just as hard or harder.

2. In order to find the median of an (unsorted) array of numbers, one ~~must~~ first sort the provided array before then retrieving the element in the middle of the array. After sorting, retrieving this element should take constant time.

   ❌

   In order to find the median of an (unsorted) array of numbers, **one can first sort the provided array** before then retrieving the element in the middle of the array. After sorting, retrieving this element should take constant time.

3. The solution A of sorting an array of numbers can be used by the solution B to find a median of numbers by using solution A and then getting the median using an index which is $\Theta(1)$ time. Since solution B can use solution A, then problem A must be just as hard or possibly harder than problem B.

   ✅

4. Sorting an array means traversing all the elements of the array and comparing with each other. Median of an array of elements can be found out by sorting half the elements of the array and print the middle element.

   ❌ Because it could be the second half.

5. Sorting an array produces an ordered array of numbers with a known length. To determine the median is a constant calculation in a sorted array, and is within $\Theta(1)$ of sorting the array, you either take the value in the middle of the sorted array with odd length, or the average of the two values in the values in the middle of an array with even length.

   ✅

6. Sorting an array be Problem A and finding the median of an array be Problem B: Problem A is just as hard or possibly harder than solving Problem B since the median of a sorted array is its middle element. So if we solve Problem A then Problem B will be solved. Which is why it is just as hard or possibly harder than solving Problem B.

   ✅

7. Suppose we have a solution S for sorting problem. Finding a median number is like sorting only half the array and take the middle one as a result, or sorting the whole array and take the middle one. Therefore it takes only half time compare to sorting an array because it only needs to use half of solution of S. This makes sorting an array as hard or possibly harder.

   ❌

8. Because the solution which can solve the problem of sorting an array of numbers can also solve the problem of finding a median of an array of numbers.

   ✅

9. Both require iterating over all array elements. In order to find the median of the array you have to see all elements in the array. All elements must be iterated over to check if an array is sorted. Therefore, sorting an array is as hard as finding a median of an array of numbers.

    ❌

10. The solution can solve the problem of sorting an array of numbers can also solve the problem of finding a median of an array of numbers.

    ✅

11. Problem of sorting an array of numbers is just as hard or possibly harder than problem of finding a median would just make the array sorted and then divide the number of elements in the array by 2.

    ❌

    finding a median of the array, it doesn't make the array sorted.

12. Finding the median number from an array requires to know the "ranking" of the number size which is essentially sorting an array and it is the minimum operation to find the median, so it is just as hard or possibly harder.

    ✅

13. You can sort the array and find the median in O(1) by looking at index $\frac{n}{2}$

    ❌

14. You must sort the array (or part of the array) to find the median.

    ❌

15. Sorting an array requires at least one loop through the array to get through the array and finding the median of an array of numbers may take at least half of the number of iterations.

    ❌

16. Sorting an array requires running through each number and placing it in order. The same process must be done in order to find the median value, or in other words, you need to sort the array to find the median, therefore the problems are just as hard as each other.

    ❌

Sets and Relations

A **Set** is a collection of distinguishable items.

- For Example S = { 1, 2, 3 }
- We can say 1 in S and 4 "not-in" S

The **Cardinality** of a Set is the number items.

- |S| = 3

This is a set containing the empty set. And the Cardinality is 1

$S = \{\emptyset\}$

The **Empty Set** is the set with no items.

- S = {}

$S = \emptyset$

Cardinality of Empty set = 0

Sometimes the empty set is written in this way

Set A is a **Subset** of set B if all the elements of set A are contained in Set B.

Set A is a **Proper Subset** of set B if A is a subset of B and A != B.

A **Power Set** is the set of all subsets.

- P(S) = { {}, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3} }
- $|P(S)| = 2^S$

$2^3 = 8$

000 {}
001 {1}
010 {2}
011 {1,2}
100 {3}
101 {3,1}
110 {3,2}
111 {3,2,1}

The **Cartesian Product** of two sets A, B is the set of all ordered pairs with first element is an element of A and the second is an element of B.

- A = {x,y} ; B = {1,2} A X B = { (x,1), (x,2), (y,1), (y,2) }
- |A x B| = |A| x |B|

$|R| = 6$, $|S| = 7$

$|R \times S| = 42$

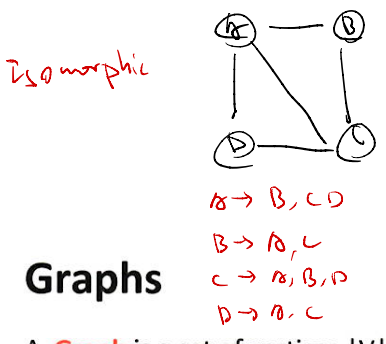
A **Relation** of two sets A, B is a subset of the Cartesian product.

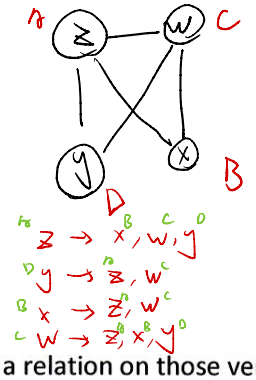
$(a,b) \in R$

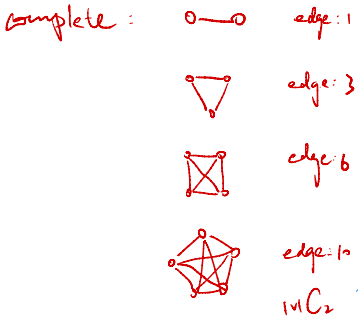
- a R b
- Reflexive a R a $(a,a) \in R$
- Transitive a R b and b R c => a R c $(a,b) \in R$, $(b,c) \in R \Rightarrow (a,c) \in R$
- Symmetric a R b -> b R a $(a,b) \in R \Rightarrow (b,a) \in R$
- Antisymmetric a R b and b R a implies a = b $(a,b) \in R$ $(b,a) \in R$
- Equivalence = Reflexive, Transitive, Symmetric --- form a partition of equivalence classes

反对称

If a related to b and b related to b only happens if b equals b







Graphs

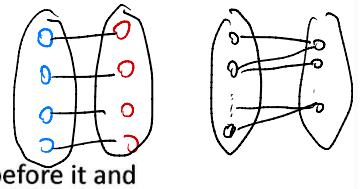
A **Graph** is a set of vertices |V| and a relation on those vertices called edges, |E|.

Graphs are **Isomorphic** if there exists a mapping from the vertices of one graph to the vertices of another graph where the edge relations are the same.

Graph G' is a **Subgraph** of G iff V' is a subset of V and E' is a subset of E

The subgraph is **induced** where E' = {(u,v) in E : u,v in V'}

A **complete** graph is a graph where every pair of vertices is connected



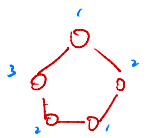
A **bipartite** graph can be partitioned into two sets where all edges go between the sets.

A **Walk** is a sequence of vertices and edges such that every vertex is incident to the edge before it and after it.

A **Path** is an open walk with no repeated vertices.

A **cycle** is closed walk where no vertex appears twice (except beginning and end)

a cycle with an odd number of vertices is not bipartite



at least edge = points

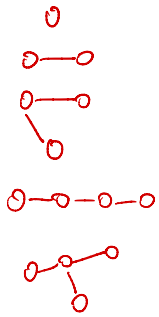
The **Degree** of a vertex is the number of edges incident to it.

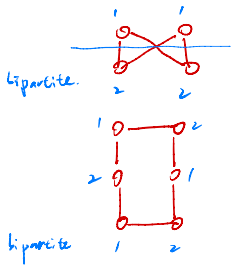
A **Tree** is a connected, acyclic graph.

$|V| = |E| + 1$

A **component** is a maximal connected subgraph.

Tree

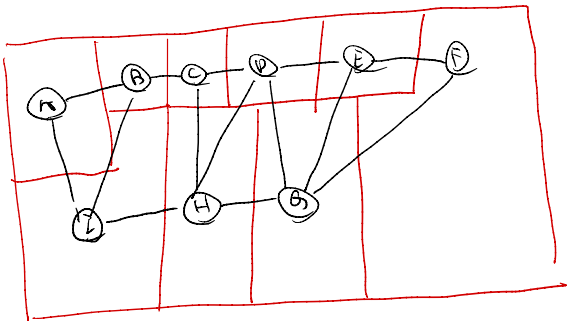




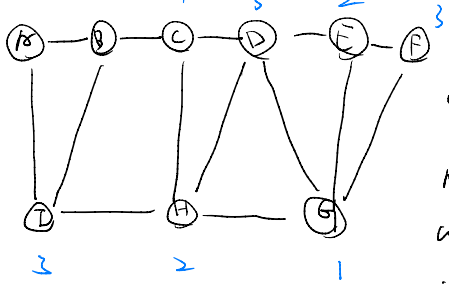
Storing Graphs

- Adjacency Matrix and Adjacency Lists
-

coloring

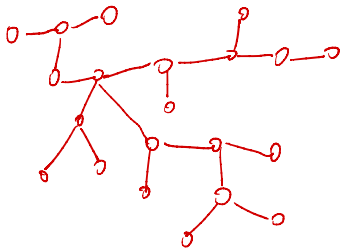


(A C G = 1) (B H E = 2) (D F I = 3)

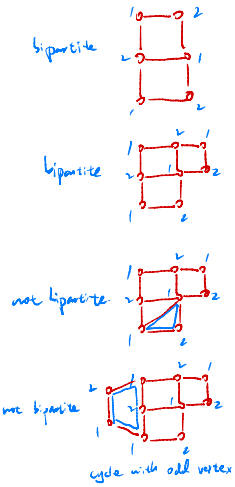


pick the smallest

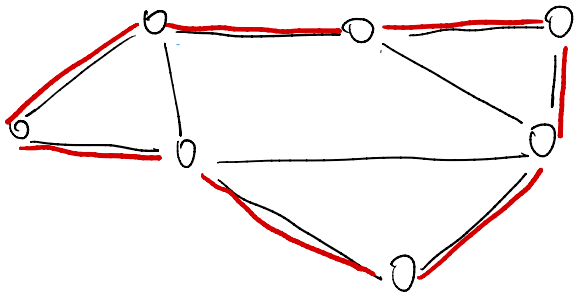
coloring a graph is NP complete, finding the largest complete subgraph is also NP complete



Trees are bipartite because no cycles.

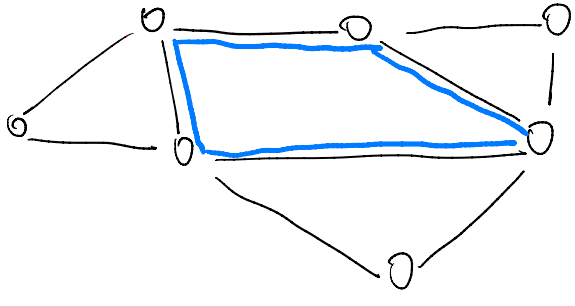


There are a lot of things that are easy to do with bipartite graphs, for instance, coloring with three or more colors is not easy. coloring a bipartite graph is linear time with a greedy algorithm.



Hamiltonian cycle

Visit each vertex exactly once (like the red)

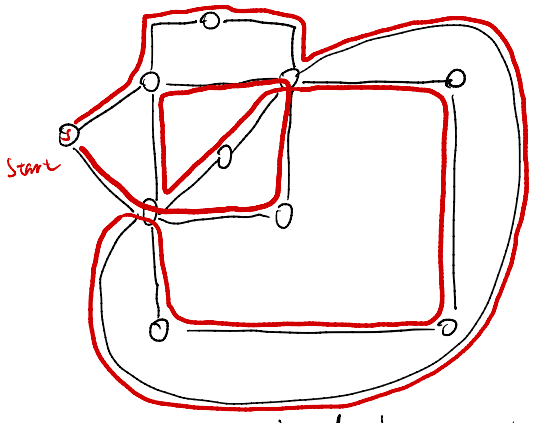


The blue is also cycle but not the hamiltonian cycle because it does not include all of the vertices of the graph

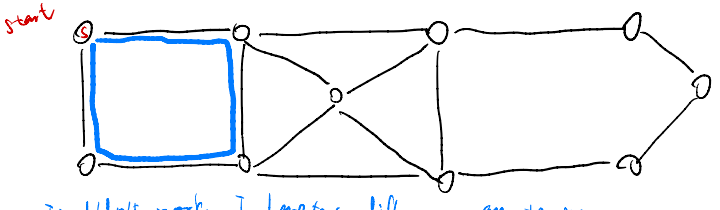
Determing if a graph has a hamiltonian cycle is also NP complete

Eurler tour

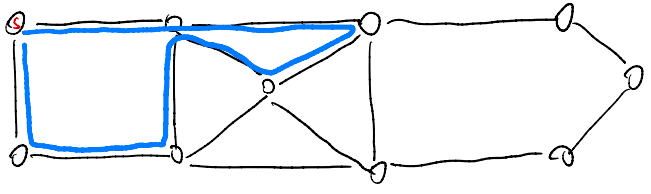
Visit each edge exactly once + end up of your starting vertex



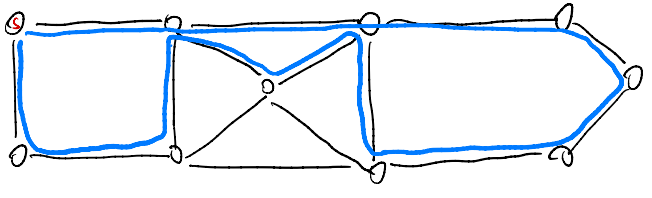
visiting all the vertices is hard but visiting all the edges is easy.



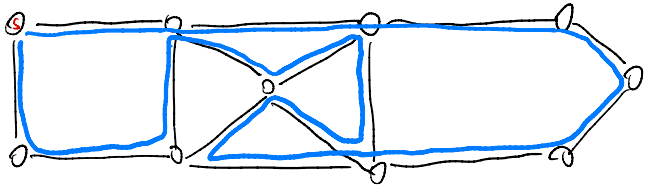
It didn't work. I have to a different way to go



This is help but I'm not done yet. so I'm going to keep following turn in different way



Almost got it but no quite

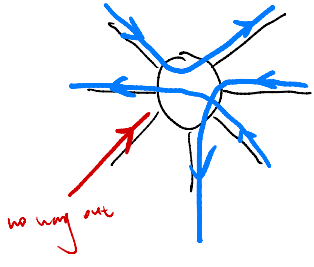


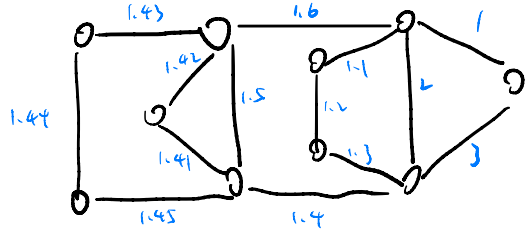
Done!

So whenever you find a path that you didn't take you break where you're going and you take that path instead and kind of add it into what you're doing.

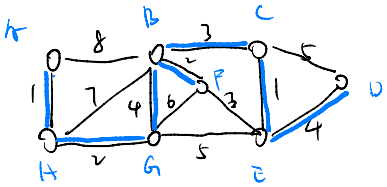
How do you know that a graph contains a Hamilton or not a hamiltonian cycle?

How do you know a graph contains an Euler tour?





The final one take all of these in order from smallest to largest and that would be your Euler tour repeat. For example. treat them as crossings, without going the crossings paths twice how could do that.



eight cities here, you want to connect them all. with minimum cost. which roads should you build?

Min Spanning Tree

Kruskal algorithm.

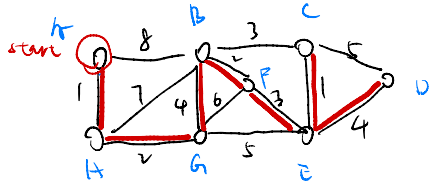
you pick the road you can build of lease cost.

1: AH. CE

2: HG. BF

3: BC FE → but this will make a cycle. so we don't need this.

4: ED. BG done. $1\times2+2\times2+3\times1+4\times2=2+4+3+8=17$



Prim's algorithm:

Prim starts from a particular place. for example we start from A. for just only can road between A and can't see like some roads above C. which road can add for minimum cost for my already exist tree of A.

A H G B F E C D $1+2+4+2+3+1+4=7+6+4=17$