**[25th May 2020, 8:30-11pm(IST)]   
[Class 1]**

Lecture Notes:

==============================

Articles/Notes:

→

<https://www.geeksforgeeks.org/understanding-time-complexity-simple-examples/>

→

<https://en.wikipedia.org/wiki/Time_complexity>

→ <https://www.hackerearth.com/practice/basic-programming/complexity-analysis/time-and-space-complexity/tutorial/>

<https://leetcode.com/articles/contains-duplicate/>

<https://www.geeksforgeeks.org/trapping-rain-water/>

Class notes:

|  |
| --- |
| // c++ / c / Java - same // Time: O(n\*n) Space : O(1) ----> O(n) -- Home work : O(n\*n) -> O(n) int arr[] = {3, 0, 2, 0, 4} int left[] = {3,3,3,3,4} -> left[i] = left\_max(i); int right[] = {4,4,4,4,4} -> right[i] = right\_max(i); // Time: O(n\*n) Space : O(1) ----> Time: O(n) int find\_water (int arr[], int n){  int total = 0;  for(int i=0;i<n;i++){  total = total + min(left\_max(arr,n,i),right\_max(arr,n,i)) - arr[i]; // O(n) -> O(1)  }  return total; } // O(n) --> O(1) int left\_max(int arr, int n, int i){  return left[i];  /\*  int mx=0;  for(int j=0;j<=i;j++){  mx = max(arr[j],mx);  }  return mx;  ?\*/} |

=============

**[27th May 2020, 8:30-11pm(IST)]**

**[Class 2: Arrays, Binary Search!]**

Articles/Notes:

→

<https://www.geeksforgeeks.org/array-data-structure/>

→

<https://www.geeksforgeeks.org/how-do-dynamic-arrays-work/>

→

<https://brilliant.org/wiki/dynamic-arrays/>

→ <https://stackoverflow.com/questions/28014541/calculating-time-complexity-of-a-dynamic-array>

Problems:

→

<https://www.geeksforgeeks.org/write-a-program-to-reverse-an-array-or-string/>

→

<https://www.geeksforgeeks.org/array-rotation/>

→

<https://www.geeksforgeeks.org/program-for-array-rotation-continued-reversal-algorithm/>

→

<https://www.geeksforgeeks.org/linear-search/>

→

<https://www.geeksforgeeks.org/binary-search/>

→

<https://www.geeksforgeeks.org/find-first-and-last-positions-of-an-element-in-a-sorted-array/>

→

[https://www.geeksforgeeks.org/count-number-of-occurrences-or-frequency-in-a-sorted-array](https://www.geeksforgeeks.org/count-number-of-occurrences-or-frequency-in-a-sorted-array/)

Visualization tool(s):

→

<https://www.cs.usfca.edu/~galles/visualization/Search.html>

Practice Problems:

[Easy-Medium]

→

<https://www.geeksforgeeks.org/check-if-a-given-number-is-a-perfect-square-using-binary-search/>

→

<https://www.geeksforgeeks.org/floor-square-root-without-using-sqrt-function-recursive/>

→

<https://codeforces.com/problemset/problem/750/A>

→

[Medium]

→

<https://www.geeksforgeeks.org/minimum-capacity-of-small-arrays-needed-to-contain-all-element-of-the-given-array/>

→

<https://www.geeksforgeeks.org/median-of-two-sorted-arrays/>

→

<https://codeforces.com/problemset/problem/1352/C>

→ Transcript/Class-Notes:

|  |
| --- |
| /\*  Array! \*/  int 32 bit, String, char    Arrays: cont. memory blocks  arr = [[],[],[],[],[],[],[]]  arr =[2,4,5,6,7,21]  1). Indexing, cont. allocation  {arr[idx]}  arr[0]=2;  arr[1]=4;  2). Cache friendly     int arr[size];  int [] arr = new int [sz];  arr[idx];   Operations on static(size fixed) array(unosorted):  Insertion:  1). End insertion: O(1)  2). Random insertion other than end = O(n);  Search:  O(n)   Operations on the dynamic array(unsorted):   Search in an array:   1). Linear Search  Check each elements  1). Usorted Array = O(n)  2). Sorted Array = O(n)  2). Binary Search:   /\* Binary Search!\*/  Binary search is a powerful divide & conquer technique to work on the problems where the given input  data show monotonicity i.e. the data is  either always increasing or decreasing ( not both ).     int firstOccurance(int arr[], int n, int X){  int l = 0;  int r = n-1;  while(l<=r){  int mid = (l+r)/2;  if((mid==0 || X > arr[mid-1]) && arr[mid]==X)   return mid;  if(X>arr[mid]) l = mid+1;  else r = mid-1;  }  int lastOccurance(int arr[], int n, int X){  int l = 0;  int r = n-1;  while(l<=r){  int mid = (l+r)/2;  if((mid==n-1 || X < arr[mid+1]) && arr[mid]==X)   return mid;  if(X>arr[mid]) l = mid+1;  else r = mid-1;  }      Problem: Given sorted array of size 'n'. Find how many times X comes in the array! arr = [2,2,2,3,4,5,6] //Brute Force: int cnt = 0; for(int i=0;i<n;i++)  if(arr[i]==X) cnt++; return cnt;  O(n)  firstOccurance  lastOccurance   cnt = lastOccurance - firstOccurance + 1;  O(logn+logn)  O(log n)  Complexity of Binary Search:  n , n/2, n/4 .... 1   O(log n)     16 -> 8 -> 4 -> 2 -> 1 = log[base2]16  15 -> 7 -> 3 -> 1 =     while( l<=r ){  //Constant time Operations  }  No of time loop runs \* O(1)  log n \* O(1)  O(log n) |

----------------

**[29th May 2020, 9:30-11pm(IST)]**

**[Class 3: Prefix Sum!]**

→ Articles/Notes/Problems:

→

<https://www.hackerrank.com/topics/prefix-sum>

→ <https://www.geeksforgeeks.org/prefix-sum-array-implementation-applications-competitive-programming/>

→

<https://www.geeksforgeeks.org/constant-time-range-add-operation-array/>

→ Transcript/Class-Notes:







|  |
| --- |
| /\*  Class 3:  Prefix Sum! \*/   /\*  Problem 1: You have an array of size 'n'. You have 'Q' queries, each query is   in the form of [L,R] where 0<= L <= R < n.  For each query you have to print the sum of all the elements of arr   from index 'L' to index 'R';    I/P: arr=[1,4,5,2,-1], n = 5  Q = {[0,1],[0,4],[3,4]}  O/P: 5, 11, 1  I/P: arr=[1,1,-2,2,-1], n = 5  Q = {[0,1],[0,4],[3,4]}  O/P: 2, 1, 1 \*/  // Solution 1: // Brute  // [0,1] // Time complexity in terms of Q, n! //O(Q \* n) void BruteSolution(int n, int arr[], int Queries[][2] ){  for(int i=0;i<Queries.size();i++){ Q times  int L = Queries[i][0];  int R = Queries[i][1];  int sum = 0;  for(int j=L;j<=R;j++) //O(n)  sum+=arr[j];   print(sum);  } } |

|  |
| --- |
| //O(n+Q) void solution(int n, int arr[], int Queries[][2] ){  int pre[n];  pre[0]=arr[0];  for(int i=1;i<n;i++) //O(n) precomputation  pre[i]=pre[i-1]+arr[i];  for(int i=0;i<Queries.size();i++){  int L = Queries[i][0];  int R = Queries[i][1];  int sum = 0;  sum = pre[R]-pre[L]+arr[L]; // O(1)  print(sum);  } }  }  /\*  Problem 2:  You have an array of size 'n' in which all the elements are initially '0'.  You have to perform Q operations on it:  Each operation is of type [L, R, D]:  add D to all the elements from index 'L' to index 'R';   Print the final array after performing all these Q operations!   arr = [0,0,0,0,0,0], n = 6;  Q = 4   [0,1,5],   [2,3,1],  [3,5,2],  [0,5,2]  [0,1,5]  [0+5,0+5,0,0,0,0]    [2,3,1]  [0+5,0+5,0+1,0+1,0,0]    [3,5,2]  [0+5,0+5,0+1,0+1+2,0+2,0+2]    [0,5,2]  [0+5+2,0+5+2,0+1+2,0+1+2+2,0+2+2,0+2+2]    [7,7,3,5,4,4]  \*/ //O(q\*n+n) //O(q\*n) void bruteforce(int n, int arr[], int Q[][3], int q ){  for(int i=0;i<q;i++){//O(Q)  int l = Q[i][0];  int r = Q[i][1];  int d = Q[i][2];  for(int j=l;j<=r;j++)//O(n)  arr[j]=arr[j] + d;  }  for(int i=0;i<n;i++)//O(n)  print(arr[i]); }  // when all array elements are zero void solution(int n, int arr[], int Q[][3], int q){  int pre[n+1];  for(int i=0;i<q;i++)  pre[i]=0;  for(int i=0;i<q;i++){// Q time  int l = Q[i][0];  int r = Q[i][1];  int d = Q[i][2];  pre[l]+=d;  pre[r+1]-=d;  }  int sum=0;  for(int i=0;i<n;i++){  sum+=pre[i];  print(sum);  }  //O(Q+n) } // when there might be non-zero elements as well void solution(int n, int arr[], int Q[][3], int q){  int pre[n+1];  for(int i=0;i<q;i++)  pre[i]=0;  for(int i=0;i<q;i++){// Q time  int l = Q[i][0];  int r = Q[i][1];  int d = Q[i][2];  pre[l]+=d;  pre[r+1]-=d;  }  int sum=0;  for(int i=0;i<n;i++){  sum+=pre[i];  print(sum+arr[i]);  }  //O(Q+n) } |
|  |
|  |

**[1st June 2020, 8:30-11pm(IST)]**

**[Class 4: Hashing & Applications]**

→ Articles/Notes:

→ <https://www.geeksforgeeks.org/hashing-set-1-introduction/>

→ <https://www.geeksforgeeks.org/index-mapping-or-trivial-hashing-with-negatives-allowed/>

→ <https://www.geeksforgeeks.org/hashing-set-2-separate-chaining/>

→ <https://www.geeksforgeeks.org/hashing-set-3-open-addressing/>

→ <https://www.geeksforgeeks.org/double-hashing/>

→ Problems:

→ <https://www.geeksforgeeks.org/count-distinct-elements-in-an-array/>

→ <https://www.geeksforgeeks.org/design-a-data-structure-that-supports-insert-delete-search-and-getrandom-in-constant-time/>

→ <https://www.geeksforgeeks.org/given-an-array-a-and-a-number-x-check-for-pair-in-a-with-sum-as-x/>

→ <https://www.geeksforgeeks.org/find-if-there-is-a-subarray-with-0-sum/>

→ <https://www.geeksforgeeks.org/find-subarray-with-given-sum-in-array-of-integers/>

→ Additional(Advanced):

→ <https://www.geeksforgeeks.org/load-factor-and-rehashing/>

→ <https://www.geeksforgeeks.org/bloom-filters-introduction-and-python-implementation/>

→ <https://www.cs.cmu.edu/~avrim/451f11/lectures/lect1004.pdf>

→Class notes:

|  |
| --- |
| /\*  Problem 1:    You have an integer array 'arr' of size 'n' and a number X.  You have to find two distinct  numbers from the array such that their sum is 'X'.    In other words:  Find two indices 'i' & 'j' such that i!=j and arr[i]+arr[j]==X    I/P: arr = {11,2,3,4,15}, X=14  O/P: Exists    I/P: arr = {11,2,3,4,15}, X=17  O/P: Exists    I/P: arr = {11,2,3,4,15}, X=21  O/P: No solution!   \*/  //O(n\*n) String BruteForce(int arr[], int X, int n){  for(int i=0;i<n;i++){  for(int j=0;j<n;j++){  if(i==j) continue;  if(arr[i]+arr[j]==X) return "YES";  }  }  return "NO"; } [11,2,3,4,15] X = 14!  //Sorting based approach!  // Sort the array  [2,3,4,11,15]  l = 0;//arr[l]=2; r = n - 1;//arr[r]=15;  curr = arr[l]+arr[r]; curr = 2+15;//17   r decrease => arr[r] will decrease => curr will decrease l increase => arr[l] will increase => curr will increase  String SortingBased(int arr[],int X, int n){  //Arrays.sort();  //sort(arr,arr+n); nlogn  int l = 0;  int r = n-1;  while(l<r){  int curr\_sum = arr[l]+arr[r];  if(curr\_sum==X) return "YES";  if(curr\_sum < X){  l++;  }  else{  r--;   }  }  //O(nlogn)  return "NO"; }  I/P: arr = {11,2,3,4,15}, X=14 O/P: Exists  arr = {2,3,4,11,15}; int l = 0; int r = 4;  iteration 1: l = 0; r = 4; curr = 2+15;//17 curr > X r--;  iteration 2: l = 0; r = 3; curr = 2+11;//13  curr < X (14) l++;   iteration 3: l = 1; r = 3; curr = 3+11;//14 X==curr return "YES";    String Solution3CPP(int arr[], int n, int X){  unordered\_set<int> hashset;//C++  for(int i=0;i<n;i++){  int curr = X - arr[i];  //C++  if(hashset.find(curr)!=hashset.end()){  return "YES";  }  //C++;  hashset.insert(arr[i]);  }  return "NO"; } String Solution3Java(int arr[], int n, int X){  HashSet<Integer> hashset = new HashSet<Integer>();//Java  for(int i=0;i<n;i++){  int curr = X - arr[i];  //Java  if(hashset.contains(curr)){  print(arr[i],curr)  return "YES"  }  //Java  hashset.add(arr[i]);  }  return "NO";   } Time = O(n) Space = O(n)   I/P: 11,2,3,4 X=4  I/P: arr = {11,3,4,15}, X=14 O/P: Exists!  i = 0; Hashset = {}; curr = X - 11 = 3  i=1; Hashset = {11} curr = 2;  i = 2 Hashset = {11,2} curr = 14- 3 ==11 print return "YES"  /\*  Problem 2:  Given an array of size 'n'. Find a subarray with sum = 0;  I/P: arr = {1,2,3,-5,6}  O/P: subarray from index 1 to 3   \*/   I/P: arr = {1,2,3,-5,6}  O/P: subarray from index 1 to 3   i=0; hashset ={} preSum = 1;  i=1; hashset = {1} preSum = 3   i=2; hashset = {1,3} preSum = 6;   i=3; hashset = {1,3,6} preSum = 1; return "YES";   I/P: arr = {2,3,-5,6}  I/P: arr = {0,2,3,-5,6}   void zeroSumSubCPP(int arr[], int n){  unordered\_set<int> hashset;  int preSum=0;  for(int i=0;i<n;i++){  preSum+=arr[i];  if(hashset.find(preSum)!=hashset.end() || preSum==0){//check if present in hashset  print("YES");  exit(0);  }  hashset.insert(preSum)  }  print("NO"); }   void zeroSumSubCPP(int arr[], int n){  map<int,int> hashmap;  int preSum=0;  for(int i=0;i<n;i++){  preSum+=arr[i];  if(preSum==0){  print(0, " to ", i);  exit(0);  }  if(hashmap.find(preSum)!=hashmap.end()){  print(hashmap[preSum]+1 , " to ", i)  //check if present in hashset  print("YES");  exit(0);  }  hashmap[preSum]=i;  }  print("NO"); }   I/P: arr = {1,2,3,-5,6}    i=0;  hasmap ={}  preSum = 1;    i=1;  hasmap={[1,0]}  preSum=3;    i=2;  hasmap={[1,0],[3,1]}  preSum=6;    i=3;  hasmap={[1,0],[3,1],[6,2]}  preSum=1;  0+1 to 3   1 to 3  /\*  Problem 3:  Given an array of size 'n' and an integer X. Find a subarray with sum = X;  I/P: arr = {1,2,3,-5,6}  O/P: subarray from index 1 to 3   \*/ |

----------------------------

**[3rd June 2020, 8:30-11pm(IST)]**

**[Class 5: Strings]**

→ Articles/Notes:

→ <https://www.geeksforgeeks.org/string-data-structure/>

→ <https://www.geeksforgeeks.org/string-data-structure/#C%20AND%20C++>

→ <https://www.geeksforgeeks.org/string-data-structure/#Java>

→ <http://www.asciitable.com/>

→ <https://en.wikipedia.org/wiki/ASCII>

→ Problems:

→ <https://www.geeksforgeeks.org/reverse-a-string-in-c-cpp-different-methods/>

→ <https://www.geeksforgeeks.org/left-rotation-right-rotation-string-2/>

→ <https://www.geeksforgeeks.org/c-program-check-given-string-palindrome/>

→ <https://www.geeksforgeeks.org/check-whether-two-strings-are-anagram-of-each-other/>

→ <https://www.geeksforgeeks.org/find-lost-element-from-a-duplicated-array/>

→ <https://www.geeksforgeeks.org/given-two-strings-find-first-string-subsequence-second/>

→ <https://www.geeksforgeeks.org/print-subsequences-string/>

→ <https://www.geeksforgeeks.org/power-set/>

→ Problems(Practice):

→ [https://www.geeksforgeeks.org/remove-minimum-number-characters-two-strings-become-anagram](https://www.geeksforgeeks.org/remove-minimum-number-characters-two-strings-become-anagram/)  
→ <https://www.geeksforgeeks.org/rearrange-characters-string-no-two-adjacent/>

→Class notes:

|  |
| --- |
| /\*  Strings! \*/  //O((2^n) \* m)  bool checkSubseq(string s1, string s2){  int n = s1.length();  int m = s2.length();  int ptr1,ptr2=0;  for(ptr1=0,ptr2=0; ptr1<n && ptr2<m; ptr1++)  {  if(s1[ptr1]==s2[ptr2])  {  print(ptr1);  ptr2++;  }  }  //O(n+m)  if(ptr2==m) return true;  return false; }  s1= "[g]eeksforgeeks" s2 = "[s]reks"  itr = 0; s1= "[g]eeksforgeeks" s2 = "[s]reks"  itr = 1; s1= "g[e]eksforgeeks" s2 = "[s]reks";  itr = 2; s1= "ge[e]ksforgeeks" s2 = "[s]reks";  itr = 3; s1= "gee[k]sforgeeks" s2 = "[s]reks";  itr = 4; s1= "geek[s]forgeeks" s2 = "[s]reks";  itr = 5; s1= "geeks[f]orgeeks" s2 = "s[r]eks";  itr = 6; s1= "geeksf[o]rgeeks" s2 = "s[r]eks";  itr = 7; s1= "geeksfo[r]geeks" s2 = "s[r]eks";  itr = 8; s1= "geeksfor[g]eeks" s2 = "sr[e]ks";  itr = 9; s1= "geeksforg[e]eks" s2 = "sr[e]ks";  itr = 10; s1= "geeksforge[e]ks" s2 = "sre[k]s";  itr = 11; s1= "geeksforgee[k]s" s2 = "sre[k]s";  itr = 12; s1= "geeksforgeek[s]" s2 = "srek[s]";  return true; |

================================

Lecture 6: Bitwise and Puzzles

===================================

Puzzle based on Bits / Bitwise operators (normally asked in interviews)

1. <https://www.geeksforgeeks.org/puzzle-31-minimum-cut-puzzle/>

2. Similar Ques: <https://www.geeksforgeeks.org/puzzle-4-pay-an-employee-using-a-gold-rod-of-7-units/>

3. generalized this question for n (gold bar of length n) : <https://www.spoj.com/problems/SILVER/>

4. <https://stackoverflow.com/questions/2589986/puzzle-find-the-minimum-number-of-weights>

Similar ques and generalized for n:

<https://www.quora.com/How-do-I-weigh-1-100-kg-with-weighing-balance-using-minimum-number-of-weights>

<http://learningroots.in/cat-and-omet/minimum-number-weights-required-using-powers-2-3/>

5. <https://www.geeksforgeeks.org/weight-heavy-ball/>

--------------------------------------------------------------

int find\_min\_cut( int n){

// n= 14 -->

int cut = 1;

int count = 0; // minm no of cuts

while(n>=cut){ // log2n

print(cut);

n = n - cut;

cut = cut\*2; // 1, 2, 4, 7

count++;

}

if(n>0) print(n);

return count; // log2n - 1 or log2n

}

========================================

# Lecture 7 : BitWise, Sorting and Misc.

Bit-Wise

<https://www.geeksforgeeks.org/program-to-find-whether-a-no-is-power-of-two/>

<https://www.geeksforgeeks.org/check-whether-k-th-bit-set-not/>

<https://www.geeksforgeeks.org/check-whether-bit-given-position-set-unset/>

<https://www.geeksforgeeks.org/set-k-th-bit-given-number/>

<https://www.geeksforgeeks.org/bits-manipulation-important-tactics/>

<https://www.geeksforgeeks.org/find-element-appears-array-every-element-appears-twice/>

Sorting / Greedy

<https://www.geeksforgeeks.org/maximise-the-number-of-toys-that-can-be-purchased-with-amount-k/>

<https://www.geeksforgeeks.org/fractional-knapsack-problem/>

Misc.

<https://www.geeksforgeeks.org/design-a-data-structure-that-supports-insert-delete-search-and-getrandom-in-constant-time/>

=================================================

**[12th June 2020, 8:30-11pm(IST)]**

**[Class 8: Recursion & Linked List]**

→ Articles/Notes:

→ <https://www.geeksforgeeks.org/recursion/>

→ <https://www.geeksforgeeks.org/tail-recursion/>

→ <https://www.geeksforgeeks.org/data-structures/linked-list/>

→ <https://algorithms.tutorialhorizon.com/least-recently-used-lru-cache-using-hashmap-and-doubly-linked-list-set-1/>

→ <https://www.geeksforgeeks.org/lru-cache-implementation/>

→ Problems:

→ <https://www.geeksforgeeks.org/sum-digit-number-using-recursion/>

→ <https://www.geeksforgeeks.org/tail-recursion-to-calculate-sum-of-array-elements/>

→ <https://www.geeksforgeeks.org/linked-list-set-2-inserting-a-node/>

→ <https://www.geeksforgeeks.org/linked-list-set-3-deleting-node/>

→ <https://www.geeksforgeeks.org/search-an-element-in-a-linked-list-iterative-and-recursive/>

→ Problems(Practice):

→ <https://www.geeksforgeeks.org/write-a-c-function-to-print-the-middle-of-the-linked-list/>

→ <https://www.geeksforgeeks.org/function-to-check-if-a-singly-linked-list-is-palindrome/>

**[15th June 2020, 8:45-10:30pm(IST)]**

**[Class 9: Linked List Problems]**

-- Sachin(1 hour 45 minutes)

→

<https://www.geeksforgeeks.org/nth-node-from-the-end-of-a-linked-list/>

→

<https://www.geeksforgeeks.org/detect-and-remove-loop-in-a-linked-list/>

→

<https://www.geeksforgeeks.org/detect-loop-in-a-linked-list/>

→

<https://www.youtube.com/watch?v=-YiQZi3mLq0>

## **[17th June 2020, 8:30-11:45pm(IST)]**

## **[Class 9: Stacks and Queues]**

-- Shashi and Nitesh

→

<https://www.geeksforgeeks.org/design-a-data-structure-for-lru-cache/>

→

[https://www.geeksforgeeks.org/stack-data-structure-introduction-program/](https://www.geeksforgeeks.org/design-a-data-structure-for-lru-cache/)

→

[https://www.geeksforgeeks.org/queue-set-1introduction-and-array-implementation/](https://www.geeksforgeeks.org/design-a-data-structure-for-lru-cache/)

→

[https://www.geeksforgeeks.org/queue-linked-list-implementation/](https://www.geeksforgeeks.org/design-a-data-structure-for-lru-cache/)

→

<https://www.geeksforgeeks.org/implement-stack-using-queue/>

→

<https://www.geeksforgeeks.org/implement-a-stack-using-single-queue/>

|  |
| --- |
| queue<int> q1, q2;  int pop1() {  if(q1.empty())  return -1;  else  {  int x = q1.front();  q1.pop();  return x;  } }   void push1(int x) {  q2.push(x);  while(!q1.empty())  {  int x = q1.front();  q1.pop();  q2.push(x);  }  swap(q1,q2);  return; }  /\*   PUSH(x): O(N)   \* Enqueque x to q2  \* Dequeue one by one all elements from q1 and enqueue them to q2  \* We just swap q1, q2    POP(x): O(1)   \* Dequeque from the q1    2nd Approach:   PUSH(x): O(1)  \* Enqueue x to q1    POP(): O(N)  \* One by one dequeue everything except the last element from q1 and enqueue them to q2 \* Dequeue the last element from q1 and return it \* swqp q1 and q2  \*/  void push2(int x) {  q1.push(x);  return; }    int pop2() {  if(q1.empty())  return -1;   while(q1.size() > 1)  {  int x = q1.front();  q1.pop();  q2.push(x);  }  int ret = q1.front();  q1.pop();  swap(q1, q2);  return ret; }    // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*     queue<int> q;   void push(int x) {  q.push(x); }   int pop() {  if(q.empty())  return -1;   int size = q.size();  for (int i = 1; i <= size-1; i++)  {  int x = q.front();  q.pop();  q.push(x);  }  int res = q.front();  q.pop();  return res; }  /\* PUSH(x):  \* enqueue x to q1   POP():  \* (size(q)-1) times: we dequeue from q1 and and then enqueue it to q1 \* dequeue it; \*/ |

→

<https://www.geeksforgeeks.org/queue-using-stacks/>

|  |
| --- |
| #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  stack<int> s1, s2;  int pop() {  if(s1.empty())  return -1;  while(s1.size() > 1)  {  int x = s1.top();  s1.pop();  s2.push(x);  }  int res = s1.top();  s1.pop();  while(!s2.empty())  {  int x = s2.top();  s2.pop();  s1.push(x);  }  return res; }   void push(int x) {  s1.push(x); }   int pop() {  int res = s1.top();  s1.pop();  return res; }   void push(int x) {  while(!s1.empty())  {  int x = s1.top();  s2.push(x);  s1.pop();  }  s1.push(x);  while(!s2.empty())  {  int x = s2.top();  s2.pop();  s1.push(x);  }  return; } |

→

<https://www.geeksforgeeks.org/next-greater-element/>

|  |
| --- |
| int a[N], r[N];  int main() {  int i, n;  cin>>n;  for(i=0;i<n;i++)  cin>>a[i];   /\*  for(i=0;i<n;i++)  {  for(j=i+1;j<n;j++)  if(a[j] > a[i])  break;  if(j<n)  r[i] = j;  else  r[i] = -1;  }  \*/   stack<int> s;  for(i=0;i<n;i++)  {  while(!s.empty())  {  if(a[s.top()] < a[i])  {  int x = s.top();  s.pop();  r[x] = i;  }  else  break;  }  s.push(i);  }  while(!s.empty())  {  int x = s.top();  s.pop();  r[x] = -1;  }   for(i=0;i<n;i++)  cout<<r[i]<<endl;  return 0; }  /\*   Find next greater element  A: {10, 5, 11, 9, 12} R: {2, 2, 4, 4, -1}  A: {10, 9, 8, 7} R: {-1, -1, -1, -1}  A: {7, 8, 9, 10} R: {1, 2, 3, -1}  \*/ |

**[19th June 2020, 8:30-11:00pm(IST)]**

**[Class 9: Linked List, Stacks and Queues]**

-- Sachin(1 hour)

→ Problems:

→ <https://www.geeksforgeeks.org/merge-two-sorted-linked-lists/>

→ <https://www.geeksforgeeks.org/merge-two-sorted-lists-place/>

→ <https://www.geeksforgeeks.org/write-a-c-function-to-print-the-middle-of-the-linked-list/>

→ Class Notes:

|  |
| --- |
| Node\* mergeLists(Node\* a, Node \*b){  if(a==NULL) return b;  if(b==NULL) return a;  Node \*dummy = NULL;  if(a->data <= b.data){  dummy = a;  dummy -> next = mergeLists(a->next,b);  }  else{  dummy = b;  dummy -> next = mergeLists(a,b->next);  }  return dummy; }  // right middle in case of two middle elements void printMiddle(Node \*head)  {   struct Node \*slow\_ptr = head;   struct Node \*fast\_ptr = head;   if (head!=NULL)   {   while (fast\_ptr != NULL && fast\_ptr->next != NULL)   {   fast\_ptr = fast\_ptr->next->next;   slow\_ptr = slow\_ptr->next;   }   printf("The middle element is [%d]\n\n", slow\_ptr->data);   }  }   // left middle in case of two middle elements void printMiddle(Node \*head)  {   struct Node \*slow\_ptr = head;   struct Node \*fast\_ptr = head;   if (head!=NULL)   {   // handle length 1 linked list as corner case  if(head->next==NULL){  printf("Middle element = %d\n",head->data);  return ;  }  bool ok = false;  while (fast\_ptr != NULL && fast\_ptr->next != NULL)   {   if(ok==true)  slow\_ptr = slow\_ptr->next;   fast\_ptr = fast\_ptr->next->next;  ok = true;  }   if(fast\_ptr!=NULL){  printf("fastP==NULL, slowP = %d", slow\_ptr->next->data);  }  else{   printf("fastP.next==NULL, slowP = %d", slow\_ptr->data);  }  printf("\n");   }  } |

-- Nitesh(1.5 hours)

→ Problems:

→ <https://www.geeksforgeeks.org/next-smaller-element/>

→<https://www.geeksforgeeks.org/largest-rectangle-under-histogram/>

→ [ttps://www.geeksforgeeks.org/sliding-window-maximum-maximum-of-all-subarrays-of-size-k/](https://www.geeksforgeeks.org/sliding-window-maximum-maximum-of-all-subarrays-of-size-k/)

→ Class Notes:

|  |
| --- |
| #include<bits/stdc++.h> using namespace std;  const long long N = 2010;   int a[N], pre[N], nex[N]; // nex[i] - pre[i] -1 , a[i]  int main() {    int n, i;  stack<int> st;  cin>>n;  for(i=0;i<n;i++)  cin>>a[i];  for(i=0;i<n;i++)  {  while((!st.empty()) && (a[st.top()] > a[i]))  {  int x = st.top();  st.pop();  nex[x] = i;  }  if(!st.empty())  pre[i] = st.top();  else  pre[i] = -1;  st.push(i);  }  while(!st.empty())  {  nex[st.top()] = n;  st.pop();  }  int ans = 0;  for(i=0;i<n;i++)  ans = max(ans, (nex[i] - pre[i] -1) \* a[i]);  cout<<ans<<endl;  return 0; }   /\*   nex[i] -> 1 + (last index which can be in rectangle)  pre[i] -> (least index which can be in recatangle) - 1  A: {6, 2, 5, 4, 5, 1, 6}  Current element is x Top of the stack is y  stack will be in increasing order. x < y -> pop y and x will become the nex smaller element of y  assume z was the nex smaller element of y  a < y  if x < y pop y, and x will be the nex smaller element of y  \*/ #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  int a[N];  int main() {  int n, k, i;  cin>>n>>k;  for(i=0;i<n;i++)  cin>>a[i];  deque<int> dq;  for(i=0;i<k;i++)  {  while((!dq.empty()) && (a[dq.back()] < a[i]))  dq.pop\_back();  dq.push\_back(i);  }  cout<<a[dq.front()]<<endl;  for(i=k;i<n;i++)  {  if(dq.front() == i-k)  dq.pop\_front();  while((!dq.empty()) && (a[dq.back()] < a[i]))  dq.pop\_back();  dq.push\_back(i);  cout<<a[dq.front()]<<endl;  }  return 0; }  /\*  dequeue:  push in back side push in front side pop from back side pop from front side   n = 6 k = 3 A: { 4, 5 , 3, 7, 4, 8}  O/P : {5, 7, 7, 8}   given a number k < n return the maximum element for all windows of size k;  last of dequeue is x current element is y  PUSH:  if x < y, then x doesn't anymore, so i will just remove x I will push y in the back side of dequeue so my dequeue will be descending order at any time  \*/ |

**[22nd June 2020, 8:30-11:00pm(IST)]**

**[Class 10: Stacks and Queues, Binary Tree]**

-- Nitesh(2.5 hour)

→ Problems:

→<https://www.geeksforgeeks.org/smallest-multiple-of-a-given-number-made-of-digits-0-and-9-only/>

|  |
| --- |
| #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  int main() {  queue<long long> q;  long long x, n;  cin>>n;  q.push(9);  while(!q.empty())  {  x = q.front();  q.pop();  if(x % n == 0)  break;  q.push((x\*10));  q.push((x\*10)+9);  }  cout<<x<<endl;  return 0; }  /\*  We are given an integer N, we need to find least X (X > 0) such that X  is divisible by N and X is made up of only digits 0 and 9. Assume that the answer X is less than 10^9   N = 5 X = 90 (900, 9000)  N = 7 X = 9009   1 10 11 100 101 110   9 90 99 900 909 990 999 9000 9009  \* Maintain a queue q \* Initially enqueue 9 to q \* Till we get answer:  pop the front element.  see if it is divisible by n, if yes, we got our answer  if no insert ("current\_element" + 0) and ("current\_element" + 9) in q.  \*/ |

→ <https://www.geeksforgeeks.org/stack-set-4-evaluation-postfix-expression/>

→<https://www.geeksforgeeks.org/stack-set-2-infix-to-postfix/>

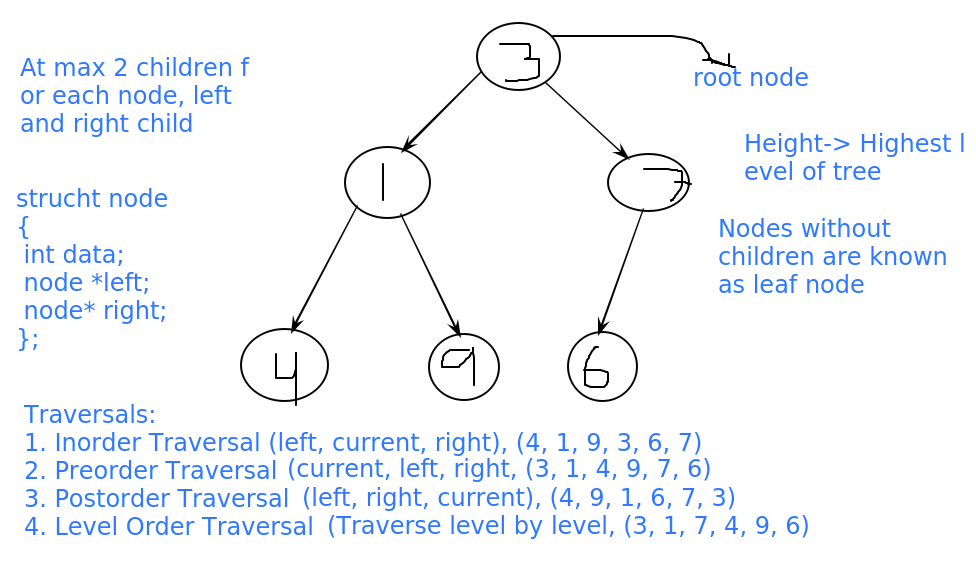
→<https://www.geeksforgeeks.org/binary-tree-data-structure/>

→<https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/>

→<https://www.geeksforgeeks.org/write-a-c-program-to-find-the-maximum-depth-or-height-of-a-tree/>

→<https://www.geeksforgeeks.org/level-order-tree-traversal/>

|  |
| --- |
| #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  struct Node {  int val;  Node \*left, \*right;  Node(int val)  {  this->val = val;  left = right = NULL;  } };  void inOrder(struct Node\* node) {  if(node == NULL)  return;  inOrder(node->left);  cout<<node->val<<endl;  inOrder(node->right);  return; }  void print\_level(struct Node\* node, int level, int desired\_level) {  if(node == NULL || level > desired\_level)  return;  if(level == desired\_level)  cout<<node->val<<endl;  print\_level(node->left, level+1, desired\_level);  print\_level(node->right, level+1, desired\_level);  return; }    void preOrder(struct Node\* node) {  if(node == NULL)  return;  cout<<node->val<<endl;  preOrder(node->left);  preOrder(node->right);  return; }  void postOrder(struct Node\* node) {  if(node == NULL)  return;  postOrder(node->left);  postOrder(node->right);  cout<<node->val<<endl;  return; }  int height(struct Node\* node) {  if(node == NULL)  return 0;  return max(height(node->left), height(node->right)) + 1; }  void levelOrder(struct Node\* node) {  queue<Node\*> q;  q.push(root);  while(!q.empty())  {  Node\* cur = q.front();  q.pop();  cout<<cur->val<<endl;  if(cur->left != NULL)  q.push(cur->left);  if(cur->right != NULL)  q.push(cur->right);  }  return; }  int main() {  struct Node\* root = new Node(1);  root->left = new Node(2);  root->right = new Node(3);  root->left->left = new Node(4);  root->left->right = new Node(5);    int h = height(root);  for(int i=1;i<=h;i++)  {  print\_level(root, 1, i);  }   cout<<endl;  cout<<endl;    levelOrder(root);   return 0; }  /\*  1  2 3  4 5   \* Maintain a queue  \* At first, insert root node  \* while queue is not empty()  pop from the front of the queue  print it  insert its left and right child in the queue    height = max\_heights(left, right) + 1  height of empty tree (NULL node) = 0  height of leaf nodes is 1   1  2  3  4  5   n -> levels  1 + 2 + 3 + .. + n = (n\*(n+1))/2 => O(n^2)   \*/ |



**[24th June 2020, 8:30-11:00pm(IST)]**

**[Class 10: Binary Search Tree]**

-- Nitesh(2.5 hour)

→ Problems:

* <https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/>
* <https://www.geeksforgeeks.org/binary-search-tree-set-2-delete/>
* <https://www.geeksforgeeks.org/advantages-of-bst-over-hash-table/>
* <https://stackoverflow.com/questions/4128546/advantages-of-binary-search-trees-over-hash-tables#:~:text=In%20summary%20if%20all%20you,preferably%20a%20self-balancing%20BST.&text=Binary%20search%20trees%20can%20be%20faster%20when%20used%20with%20string%20keys>
* <https://afteracademy.com/blog/binary-search-tree-vs-hash-table>
* <https://www.geeksforgeeks.org/inorder-successor-in-binary-search-tree/>
* <https://www.geeksforgeeks.org/inorder-predecessor-successor-given-key-bst/>

|  |
| --- |
| #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  struct Node {  int val;  Node \*left, \*right;  Node(int val)  {  this->val = val;  left = right = NULL;  } };   // If my next call to insert function is from if condition, I will get the node, which should be inserted in the left // part of the current node. struct Node\* insert(struct Node\* node, int x) {  if(node == NULL)  return new Node(x);   if(x < node->val)  node->left = insert(node->left, x);  else if(x > node->val)  node->right = insert(node->right, x);   return node; }  void insert(struct Node\* node, int x) {  if(node == NULL)  return;   if(x < node->val)  {  if(node->left == NULL)  node->left = new Node(x);  else  insert(node->left, x);  }  else if(x > node->val)  {  if(node->right == NULL)  node->right = new Node(x);  else  insert(node->right, x);  }  return; }   struct Node\* search(struct Node\* node, int x) {  if(node == NULL)  return NULL;  if(node->val == x)  return node;  else if(x < node->val)  return search(node->left, x);  else  return search(node->right, x); }  int main() {  struct Node\* root = new Node(1);  root->left = new Node(2);  root->right = new Node(3);    insert(root, 4);  return 0; }   /\*    In a binary search tree (BST),     \* It is a binary tree, every node has atmax 2 children  \* All the elements of the left part of the node, are smaller than or equal to the node  \* All the elements of the right part of the node, are greater than or equal to the node  \* Both left and right subtree of the node are also, binary search tree.     Procedure of inserting a number (x) in a BST:   \* Start from root, compare the given number x with the current node:  if x > current\_node:  insert x in right part of current node  else  insert x in left part of current node    Procedure for searching a number (x) in a BST:   \* Start from root, compare the given number x with the current node:  if x == current\_node:  return current node   else if x > current\_node:  search x in right part of current node  else  serarch x in left part of current node     \* Inorder Traversal of BST, gives its elements in sorted order.    \* Inorder successor of a given node is the node which has value just greater than the value at given node     Procedure for deleting a number (x) in a BST:  if x is a leaf node:  Remove it  else if x has 1 child:  Remove x, and attach its child to its parent  else (x has 2 children):  swap the value of x, with its inorder successor, and remove inorder successor.    Balanced Binary Search Tree \*/ |

# **Lecture By Shashi: Binary Trees , 26th June**

<https://www.geeksforgeeks.org/search-a-node-in-binary-tree/>

<https://www.geeksforgeeks.org/print-path-root-given-node-binary-tree/>

<https://www.geeksforgeeks.org/given-a-binary-tree-print-all-root-to-leaf-paths/>

<https://www.geeksforgeeks.org/root-to-leaf-path-sum-equal-to-a-given-number/>

<https://www.geeksforgeeks.org/find-the-maximum-sum-path-in-a-binary-tree/?ref=rp>

<https://www.geeksforgeeks.org/sum-nodes-longest-path-root-leaf-node/>

|  |
| --- |
| void preorder(node \*root){  if(root==null) return;  cout<<root->data;  preorder(root->left);  preorder(root->right); }  bool flag = false; void preorder(node \*root, int x){  if(root==null) return;  if(root->data==x) {  flag = true; return;  }  preorder(root->left);  preorder(root->right); }  bool find(node \*root, int x){  if(root==null) return false;  if(root->data==x) return true;  bool ans = find(root->left,x);  if(ans==true) return true;  ans = find(root->right,x);  return ans; }  bool find(node \*root, int x){  if(root==null) return false;  if(root->data==x) return true;  bool ans = find(root->left,x);  if(ans==true) return true;  return find(root->right,x); }  bool find(node \*root, int x){  if(root==null) return false;  if(root->data==x) return true;  return ( find(root->left,x) || find(root->right,x) ); }  find(1,7) = true;  =================================  Path from root node to a given node  1  2 6 7  7 2 1 bool printPath(node \*root, int x){  if(root==null) return false;  if(root->data==x) {  cout<<root->data;return true; // leaf  }  if (find(root->left,x) || find(root->right,x)){  cout<<root->data;return true; // non-leaf  }  return false; }   1  2 3 6 7 20  1 2 7  s = ""; s="1 2 7" 1 2 7 void printPath(node \*root, int x, string &s){  if(root==null) return;  s = s+ " "+ to\_string(root->data);  if(root->data==x) {  print(s);  s.erase(s.length()-1);  return;  }  printPath(root->left,x,s);  printPath(root->right,x,s);  s.erase(s.length()-1); }  ===============================  1  2 3 6 7 20  1 2 6 1 2 7 1 3 20 print all root to leaf paths  f(6, 2) f(7, 2) 1 2 6 path[]={1, 2, 7} void printPathsRecur(node\* node, int path[], int pathLen) {  if (node == NULL) return;  path[pathLen] = node->data;  pathLen++; // 1 -> 2 -> 3  if (node->left == NULL && node->right == NULL) // is\_leaf  {  printArray(path, pathLen);  }  else  {  printPathsRecur(node->left, path, pathLen); // f(6,2)  printPathsRecur(node->right, path, pathLen); // f(7,2)  } }  void printPath(node \*root, int x, string &s){  if(root==null) return;  s = s+ " "+ to\_string(root->data);  if(root->left==null && root->right==null) {  print(s);  }  else{  printPath(root->left,x,s);  printPath(root->right,x,s); }  s.erase(s.length()-1); }   1  2 3 6 7 20  maxSUmPath = 24  sum = 4 sum = 20 - false sum = 10 - true  f(1,10)  f(2,9) f(6,7) f(7,7)  bool f(root, int sum){ if(sum ==0) return 1; if(root==null) return 0; sum = sum - root->data; // 9 - 2 = 7 if(sum==0) return true; return ( f(root->left, sum) || f(root->right, sum) ); }  Finding the maximum sum path   1  2 3 6 7 20  f(1,0,0) f(2,1,0) f(6,3,0)= curr\_sum = 9, ans = 9 f(7,3,9) = curr\_sum = 10, ans = 10 f(3,1,10) = curr\_sum = 4, ans = 10 f(20,4,10) = curr\_sum = 24, ans = 24  int main(){  int ans =0;  maxSUmPath(root,0,ans);  cout<<ans; } // int ans = void maxSUmPath(node \*root, int curr\_sum, int &ans){ if(root==null) return; curr\_sum = curr\_sum + root->data; if(root->left==null && root->right==null) // check the leaf node ans = max(ans, curr\_sum); maxSUmPath(root->left, curr\_sum, ans); maxSUmPath(root->right, curr\_sum, ans); }  // ans = max(ans, curr\_sum); if(curr\_sum>ans){  ans = curr\_sum;  ans\_node = root; }   // the path can be till any non-leaf node also void maxSUmPath(node \*root, int curr\_sum, int &ans){ if(root==null) return; curr\_sum = curr\_sum + root->data; ans = max(ans, curr\_sum); maxSUmPath(root->left, curr\_sum, ans); maxSUmPath(root->right, curr\_sum, ans); }  public static int maxsum(Nodes root){  if(root==null){  return 0;  }  int maxsum\_ils=maxsum(root.left);  int maxsum\_irs=maxsum(root.right);  return Math.max( maxsum\_ilsr, maxsum\_irs) + root.data);  }   ================== print max\_sum\_path 1. mark that (leaf) node, 2. print the path from root to that node ============ |

**[29th June 2020, 8:30-11:00pm(IST)]**

**[Class 10: Binary Heaps]**

-- Nitesh(2.5 hour)

→ Problems:

* <https://www.geeksforgeeks.org/binary-heap/>
* <https://www.geeksforgeeks.org/applications-of-heap-data-structure/?ref=rp>
* <https://www.geeksforgeeks.org/array-representation-of-binary-heap/>
* <https://www.geeksforgeeks.org/insertion-and-deletion-in-heaps/>
* <https://www.geeksforgeeks.org/time-complexity-of-building-a-heap/?ref=rp>
* <https://www.geeksforgeeks.org/heap-sort/>
* <https://www.geeksforgeeks.org/merge-two-binary-max-heaps/>
* <https://www.geeksforgeeks.org/k-largestor-smallest-elements-in-an-array/?ref=rp>
* <https://www.geeksforgeeks.org/merge-k-sorted-arrays/>
* <https://www.geeksforgeeks.org/nearly-sorted-algorithm/>
* <https://www.geeksforgeeks.org/median-of-stream-of-integers-running-integers/>

|  |
| --- |
| #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  int a[N], n;   void heapify(int i) {  int idx = i, lc = 2\*i+1, rc = lc+1;  if(lc < n && a[idx]<a[lc])  idx = lc;  if(rc < n && a[idx]<a[rc])  idx = rc;  if(idx != i)  {  swap(a[idx], a[i]);  heapify(idx);  }  return; }  void build\_heap() {  for(int i=((n-1)-1)/2;i>=0;i--)  heapify(i); }  int extract\_max() {  int ret = a[0];  a[0] = a[n-1];  n--;  heapify(0);  return ret; }  void heapify\_2(int i) {  int parent = (i-1)/2;  if(i>0 && a[parent]<a[i])  {  swap(a[i], a[parent]);  heapify\_2(parent);  } }  void insert(int x) {  n++;  a[n-1] = x;  heapify\_2(n-1); }  /\*  O(n\*k\*log(k))  map<int, int> mp; int pt[k], a[k][n]; set<int> st;  for(i=0;i<k;i++)  st.insert(a[i][pt[i]]), mp[a[i][pt[i]]] = i, pt[i]++;  while(!st.empty()) {  int x = \*(st.begin());  cout<<x<<endl;  st.erase(st.begin());  int i = mp[x];  mp.erase(i);  if(pt[i] < n)  st.insert(a[i][pt[i]]), mp[a[i]][pt[i]] = i, pt[i]++;  }  \*/  int main() {  int i;  cin>>n;  for(i=0;i<n;i++)  cin>>a[i];  build\_heap();  for(i=0;i<n;i++)  cout<<a[i]<<endl;  cout<<endl;   cout<<extract\_max()<<endl;  cout<<endl;  for(i=0;i<n;i++)  cout<<a[i]<<endl;  return 0; }  /\*   A complete binary tree, has all levels filled, if you add a new node in it, then its height will increase   Binary heap:   It is an almost complete binary tree (last level has nodes in the left side)  if it has height h, then (first h-1 levels are completely filled, last level is filled from the left side).   Each node should follow the heap property:   if it is a max-heap, than, value at the node should be greater than the values at its childs  if it is a min-heap, than, value at the node should be smaller than the values at its childs    Max-heap, Min-heap   Representation using array (A):   \* Value at the root is stored at A[0].  \* Left child of the ith node is stored at index 2\*i+1  \* Right child of the ith node is stored at index 2\*i+2  \* For a given index i, its parent is stored at index (i-1)/2.  \* For storing n values, you need an array of size n.    Given a max-heap, insert the given element x in it such that, after insertion also, it remains max-heap   Extract\_Max:  Removes the maximum value from the heap and return it.   Given an array, convert it in max-heap and print the resultant array   Insertion/Deletion: O(logn) Maximum element: O(1)  Heap Sort  Given 2 max-heaps, merge them  Given 2 input arrays, return the output array representing max heap.   k largest elements of the given array:  {23, 5, 2 , 6, 233} k = 2 => 233, 23   Sort a nearly sorted array:  Given an array, and a number k, this array is nearly sorted: O(N \* Log(k))     I/P: 2, 1, 4, 3, 5 k=2  O/P: 1, 2, 3, 4, 5     Each element of the given array, is displaced (from correct positions) by atmax k indexes   Merge k sorted arrays (all arrays have size n): O(n\*k\*log(n\*k)) O(n\*k\*log(k))    k = 3, n = 3    1st: [ 2, 6, 8]   2nd: [ 1, 5, 9]    3rd: [ 4, 10, 12]   O/P: [1, 2, 4, 5, 6, 8, 9, 10, 12] (n\*k)      Find the running median    Median: Sort the given numbers, after sorting, median is the middle element. If number of elements is odd,  we will get a middle element ( for 5, 3 is middle element)  if number of elements is even, median will be the average of 2 middle elements. (for eg, 6, (a[3] + a[4])/2 )     5 7 4 9    4 5 7 9    5 6 5 6   if i have n (even) elements, avg(max(minimum n/2 elements), min(maximum n/2 elements))   (median)  max-heap(of minimum n/2 elements) min-heap(of maximum n/2 elements)    1 4 6 7 9 10 8 11  \*/ |

1st July: Lecture By Shashi, Tree Last Lecture

<https://www.geeksforgeeks.org/get-level-of-a-node-in-a-binary-tree/>

<https://www.geeksforgeeks.org/print-nodes-at-k-distance-from-root/>

<https://www.geeksforgeeks.org/get-level-node-binary-tree-iterative-approach/>

<https://www.geeksforgeeks.org/print-levels-nodes-binary-tree/>

<https://www.geeksforgeeks.org/maximum-width-of-a-binary-tree/>

<https://www.geeksforgeeks.org/print-a-binary-tree-in-vertical-order-set-3-using-level-order-traversal/>

<https://www.geeksforgeeks.org/print-binary-tree-vertical-order-set-2/>

<https://www.geeksforgeeks.org/print-left-view-binary-tree/>

<https://www.geeksforgeeks.org/print-right-view-binary-tree-2/>

<https://www.geeksforgeeks.org/print-nodes-top-view-binary-tree/>

<https://www.geeksforgeeks.org/bottom-view-binary-tree/>

|  |
| --- |
| void preorder(node \*root, int level){ if(root==null) return; cout<<root->data<<" "<<level; preorder(root->left, level+1); preorder(root->right, level+1); }  Time: O(n) Space: O(h)  log2n <=h<= n : Binary Trees   1  2 10(1,1) 8 9(2,0) 20(2,2)  15   1 0  2 1  10 1  8 2  9 2  20 2  15 3   1 0  2 1  8 2  9 2   f(1,0)-> f(2,1) -> f(9,2)  Level Order Traversal (BFS): Queue   void bfs(node \*root){  queue<pair<node\*,int> > q;  q.push\_back(make\_pair(root,0));  //Pair pair = new Pair(roo,0);  while(!q.empty()){  pair<node\*,int> p = q.front(); q.pop();  node \*node = p.first;  int level = p.second;  cout<<node->data<<" "<<level;  if(node->left!=null){  q.push\_back(node->left,level+1);  }  if(node->right!=null){  q.push\_back(node->right,level+1);  }  } }  Time: O(n) SPace: O(width) - maxm no of nodes at any given level   1<=w<=n  ================  print all the nodes which are at k distance from root node  void bfs(node \*root, int k){  queue<pair<node\*,int> > q;  q.push\_back(make\_pair(root,0));  //Pair pair = new Pair(roo,0);  while(!q.empty()){  pair<node\*,int> p = q.front(); q.pop();  node \*node = p.first;  int level = p.second;  if(level==k)  cout<<node->data<<" "<<level;  if(node->left!=null){  q.push\_back(node->left,level+1);  }  if(node->right!=null){  q.push\_back(node->right,level+1);  }  } }  ====================  printing hd  void preorder(node \*root, int hd){ if(root==null) return; cout<<root->data<<" "<<hd; preorder(root->left, hd-1); preorder(root->right, hd+1); }  void bfs(node \*root, int k){  queue<pair<node\*,int> > q;  q.push\_back(make\_pair(root,0));  //Pair pair = new Pair(roo,0);  while(!q.empty()){  pair<node\*,int> p = q.front(); q.pop();  node \*node = p.first;  int hd = p.second;  cout<<node->data<<" "<<hd;  if(node->left!=null){  q.push\_back(node->left,hd-1);  }  if(node->right!=null){  q.push\_back(node->right,hd+1);  }  } }   ====== print node data, level and hd  void preorder(node \*root,int level, int hd){ if(root==null) return; cout<<root->data<<" "<<level<<" "<<hd; preorder(root->left,level+1, hd-1); preorder(root->right,level+1, hd+1); }  class Triple{  node }   void bfs(node \*root, int k){  queue<pair<node\*,int> > q;  q.push\_back(make\_pair(root,0));  //Pair pair = new Pair(roo,0);  while(!q.empty()){  pair<node\*,int> p = q.front(); q.pop();  node \*node = p.first;  int hd = p.second;  if(node->data==a)  cout<<node->data<<" "<<hd;  if(node->left!=null){  q.push\_back(node->left,hd-1);  }  if(node->right!=null){  q.push\_back(node->right,hd+1);  }  } } =================== 200 - CTCI  x- Level y - hd Amazon  pair<int,int> find\_corordinates(node \*root, int a){  }  ============  Left View :  void bfs(node \*root){  queue<pair<node\*,int> > q;  q.push\_back(make\_pair(root,0));  //Pair pair = new Pair(roo,0);  int prev\_level = -1; // 0 --> 1  while(!q.empty()){  pair<node\*,int> p = q.front(); q.pop();  node \*node = p.first;  int level = p.second;  if(prev\_level<level){ // 1 1  cout<<node->data; // 1 , 2  prev\_level = level;  }  if(node->left!=null){  q.push\_back(node->left,level+1);  }  if(node->right!=null){  q.push\_back(node->right,level+1);  }  } }   1 2 5  4 6  8   level data   0 1  1 5  2 6   =======================  Right view  void bfs(node \*root){  queue<pair<node\*,int> > q;  q.push\_back(make\_pair(root,0));  //Pair pair = new Pair(roo,0);  int prev\_level = -1; // 0 --> 1  while(!q.empty()){  pair<node\*,int> p = q.front(); q.pop();  node \*node = p.first;  int level = p.second;  if(prev\_level<level){ // 1 1  cout<<node->data; // 1 , 2  prev\_level = level;  }  if(node->right!=null){  q.push\_back(node->right,level+1);  }  if(node->left!=null){  q.push\_back(node->left,level+1);  }  } }   Top View: hd bottom view : hd  =========================================== |

**[3rd July 2020, 8:45-11:15pm(IST)]**

**[Class 12: Graph Theory (BFS, DFS)]**

-- Nitesh(2.5 hour)

→ Problems:

* <https://www.geeksforgeeks.org/graph-and-its-representations/>
* <https://study.com/academy/lesson/graphs-in-discrete-math-definition-types-uses.html>
* <https://www.geeksforgeeks.org/connected-components-in-an-undirected-graph/>
* <https://en.wikipedia.org/wiki/Directed_acyclic_graph>
* <https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/>
* <https://www.geeksforgeeks.org/tree-back-edge-and-cross-edges-in-dfs-of-graph/>
* <https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>
* <https://www.geeksforgeeks.org/program-to-count-number-of-connected-components-in-an-undirected-graph/>

|  |
| --- |
| #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  vector<int> v[N]; int vis[N];  void bfs(int s) {  queue<int> q;  vis[s] = 1;  q.push(s);  while(!q.empty())  {  int cur = q.front();  q.pop();  cout<<cur<<endl;  for(int i=0;i<v[cur].size();i++)  {  int nxt = v[cur][i];  if(vis[nxt] == 0)  q.push(nxt), vis[nxt]=1;  }  }  return; }  void dfs(int s) {  vis[s] = 1;  cout<<s<<endl;  for(int i=0;i<v[s].size();i++)  {  int nxt = v[s][i];  if(vis[nxt] == 0)  dfs(nxt);  }  return; }   int main() {  int n, i, m, x, y, ans=0;  cin>>n>>m;  for(i=0;i<m;i++)  {  cin>>x>>y;  v[x].push\_back(y);  v[y].push\_back(x); // for directed graphs, remove this statement  }  for(i=0;i<n;i++)  {  if(vis[i] == 0)  dfs(i), ans++;  }  cout<<ans<<endl; // Number of components in the given graph  return 0; } |

**[6th July 2020, 8:30-11:pm(IST)]**

**[Class 12: Graph Theory Problems (BFS, DFS, Topological Sorting)]**

-- Nitesh(2.5 hour)

→ Problems:

* <https://www.geeksforgeeks.org/find-number-of-islands/>
* <https://www.geeksforgeeks.org/snake-ladder-problem-2/>
* <https://www.geeksforgeeks.org/topological-sorting/>
* <https://www.geeksforgeeks.org/topological-sorting-indegree-based-solution/>
* <https://www.geeksforgeeks.org/topological-sorting-using-departure-time-of-vertex/>
* <https://www.geeksforgeeks.org/find-the-ordering-of-tasks-from-given-dependencies/>
* <https://www.geeksforgeeks.org/find-whether-it-is-possible-to-finish-all-tasks-or-not-from-given-dependencies/>
* <https://www.geeksforgeeks.org/word-ladder-length-of-shortest-chain-to-reach-a-target-word/>
* <https://www.geeksforgeeks.org/minimum-time-required-so-that-all-oranges-become-rotten/>
* <https://www.geeksforgeeks.org/given-sorted-dictionary-find-precedence-characters/>
* <https://www.geeksforgeeks.org/level-node-tree-source-node-using-bfs/>
* <https://www.geeksforgeeks.org/shortest-path-in-a-binary-maze/>
* <https://www.geeksforgeeks.org/find-minimum-numbers-moves-needed-move-one-cell-matrix-another/>
* <https://www.geeksforgeeks.org/minimum-steps-reach-target-knight/>
* <https://www.geeksforgeeks.org/minimum-number-operation-required-convert-number-x-y/>
* <https://www.geeksforgeeks.org/find-whether-path-two-cells-matrix/>
* <https://www.geeksforgeeks.org/printing-pre-and-post-visited-times-in-dfs-of-a-graph/>

|  |
| --- |
| #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  vector<int> v[N]; int deg[N]; queue<int> q; // in undirected graph degree of a node is the number of edges it is connected to. // indegree of a node in directed graph is the number of edges that ends to that node. // outdegree of a node in directed graph is the number of edges that starts from that node.  int main() {  int n, i, m, x, y;  cin>>n>>m;  for(i=0;i<m;i++)  {  cin>>x>>y;  v[x].push\_back(y);  deg[y]++;  }  for(i=0;i<n;i++)  if(deg[i]==0)  q.push(i);   while(!q.empty())  {  int cur = q.front();  cout<<cur<<endl;  q.pop();  for(i=0;i<v[i].size();i++)  {  int nxt = v[i][j];  deg[nxt]--;  if(deg[nxt] == 0)  q.push(nxt);  }  }  for(i=0;i<n;i++)  if(deg[i] != 0)  break;  if(i<n)  cout<<"Cycle is present"<<endl;   return 0; }  #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  bool a[N][N], vis[N][N]; int dx[8] = {-1, 0, 1, -1, 1, -1, 0, 1}; int dy[8] = {-1, -1, -1, 0, 0, 1, 1, 1};  void dfs(int x, int y) {  vis[x][y] = 1;  for(int i=0;i<8;i++)  {  int nx = x+dx[i], ny = y+dy[i];  if(vis[nx][ny] == 0 && a[nx][ny] == 1)  dfs(nx, ny);  }  return; }   int main() {  int i, j, n, ans = 0;  cin>>n;  for(i=1;i<=n;i++) for(j=1;j<=n;j++) cin>>a[i][j];  for(i=1;i<=n;i++) for(j=1;j<=n;j++) if(vis[i][j] == 0 && a[i][j] == 1) dfs(i,j), ans++;  cout<<ans<<endl;  return 0; }  /\*  dfs on the given DAG, and maintain the out times of nodes.  sort the nodes in the order of decreasing out times, this order is a topological order.  higher out time means that node is finished after other nodes \*/ |

**[10th July 2020, 8:30-11:pm(IST)](Sachin)**

**[Class 13: Graphs, DSU]**

[First Problem]

You are given 2 strings s1 and s2 of the same length 'N', and you are given 'Q' constraints on s2.

Each constraint is in the form of [L[i], R[i]] i.e. you can only swap elements at the indices L[i] with R[i] any number of times!

You have can swap the elements in s2 as per these Q constraints any number of times!

[Note: You cannot swap 2 indices if they are not mentioned in queries]

You have to check whether it is possible to make s2 equal to s1!

Example:

I/P: s1="abcde", s2="ebadc", Q={[0,2],[2,4]}

O/P: True

Explanation:

You can swap index '0' & index '2' and then index '2' & index '4' in s2 and make it equal to s1="abcde"

Solution:

|  |
| --- |
| vector<int> adj[MAX]; bool vis[MAX];// all the values 'false' string component1\_string, component1\_string; void dfs(int curr){  vis[curr]=true;  component1\_string += s1[curr];  component2\_string += s2[curr];  for(auto next:adj[curr])  {  if(vis[next]==false)  dfs(next);  } } bool solve(int n, string s1, string s2, int q, int Q[][2]){//q = number of constraints  for(int i=0;i<q;i++){  int v1 = Q[i][0];  int v2 = Q[i][1];  // add edge  adj[v1].push\_back(v2);  adj[v2].push\_back(v1);  }  int total\_steps=0;  for(int i=0;i<n;i++){  if(vis[i]==true) continue;  component1\_string = "";  component2\_string = "";  dfs(i);  if(checkAnagrams(component2\_string,component2\_string)==false)   total\_steps+=minChangesToMakeAnagram(component2\_string,component2\_string);  }  return true; } |

[Contents]

1. <https://www.geeksforgeeks.org/union-find/>

2. <https://www.geeksforgeeks.org/union-find-algorithm-set-2-union-by-rank/>

3. <https://cs.stackexchange.com/questions/23179/if-all-edges-are-of-equal-weight-can-one-use-bfs-to-obtain-a-minimal-spanning-t>

4. <https://en.wikipedia.org/wiki/Minimum_spanning_tree>

[Complexity Proofs]

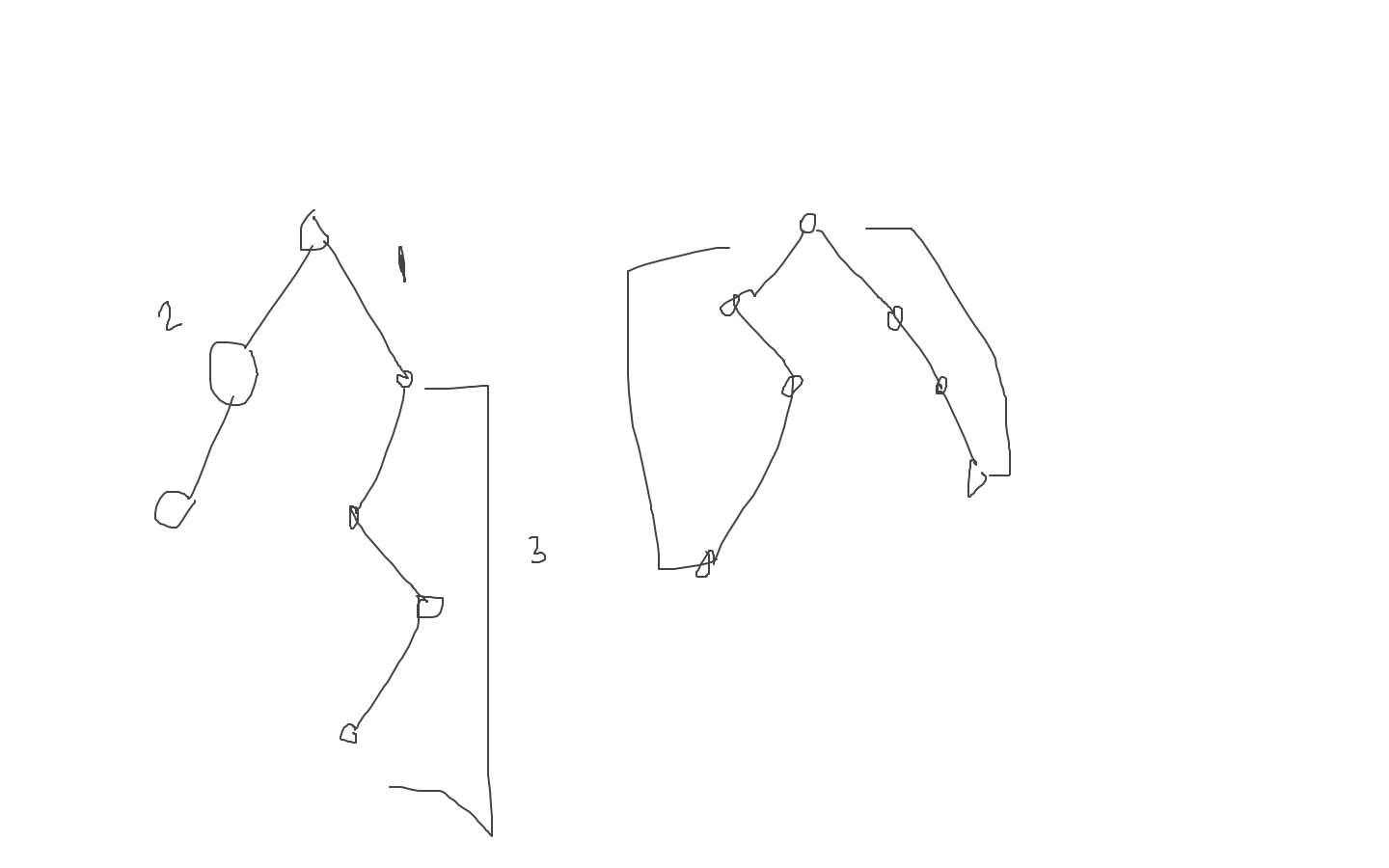
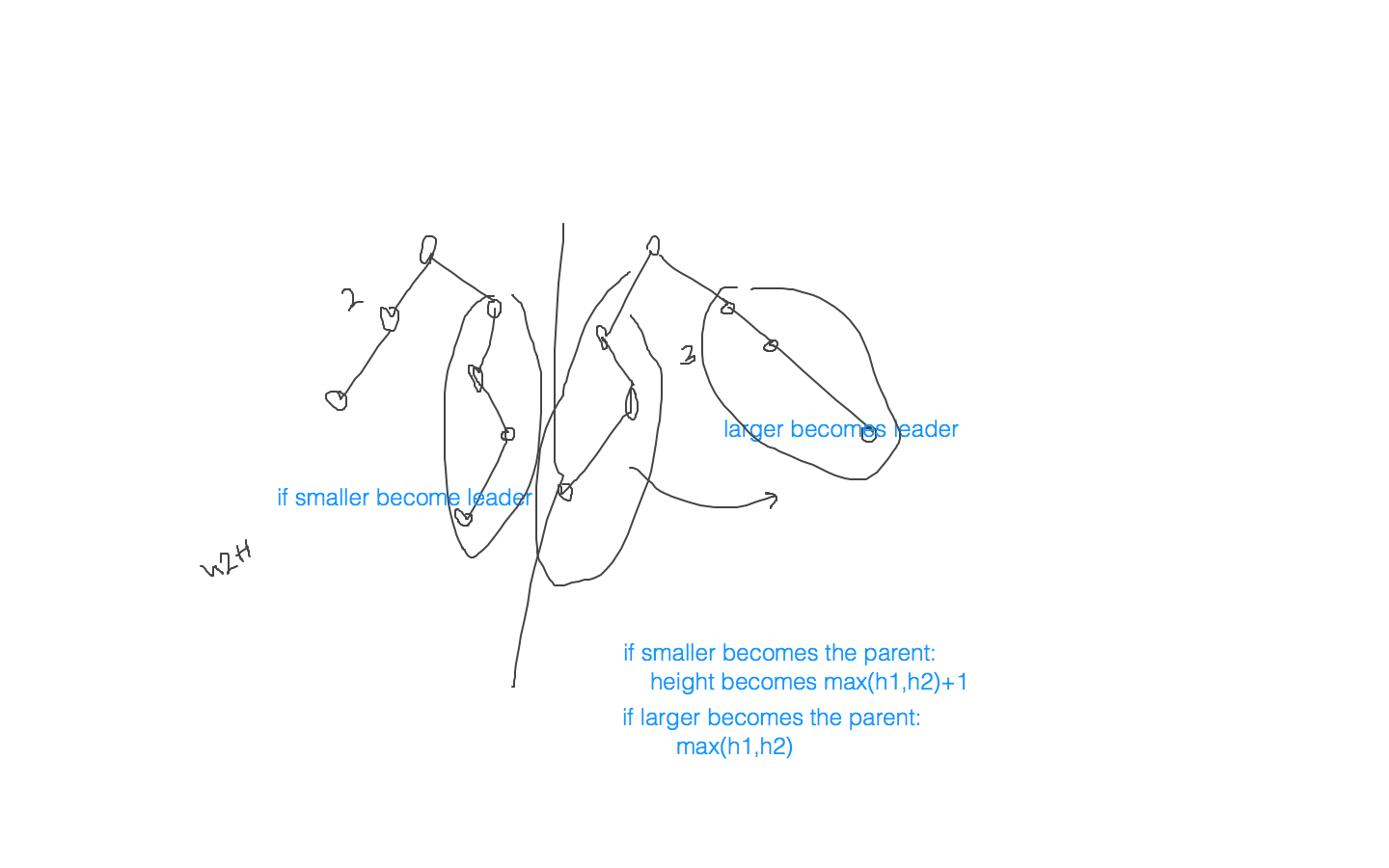
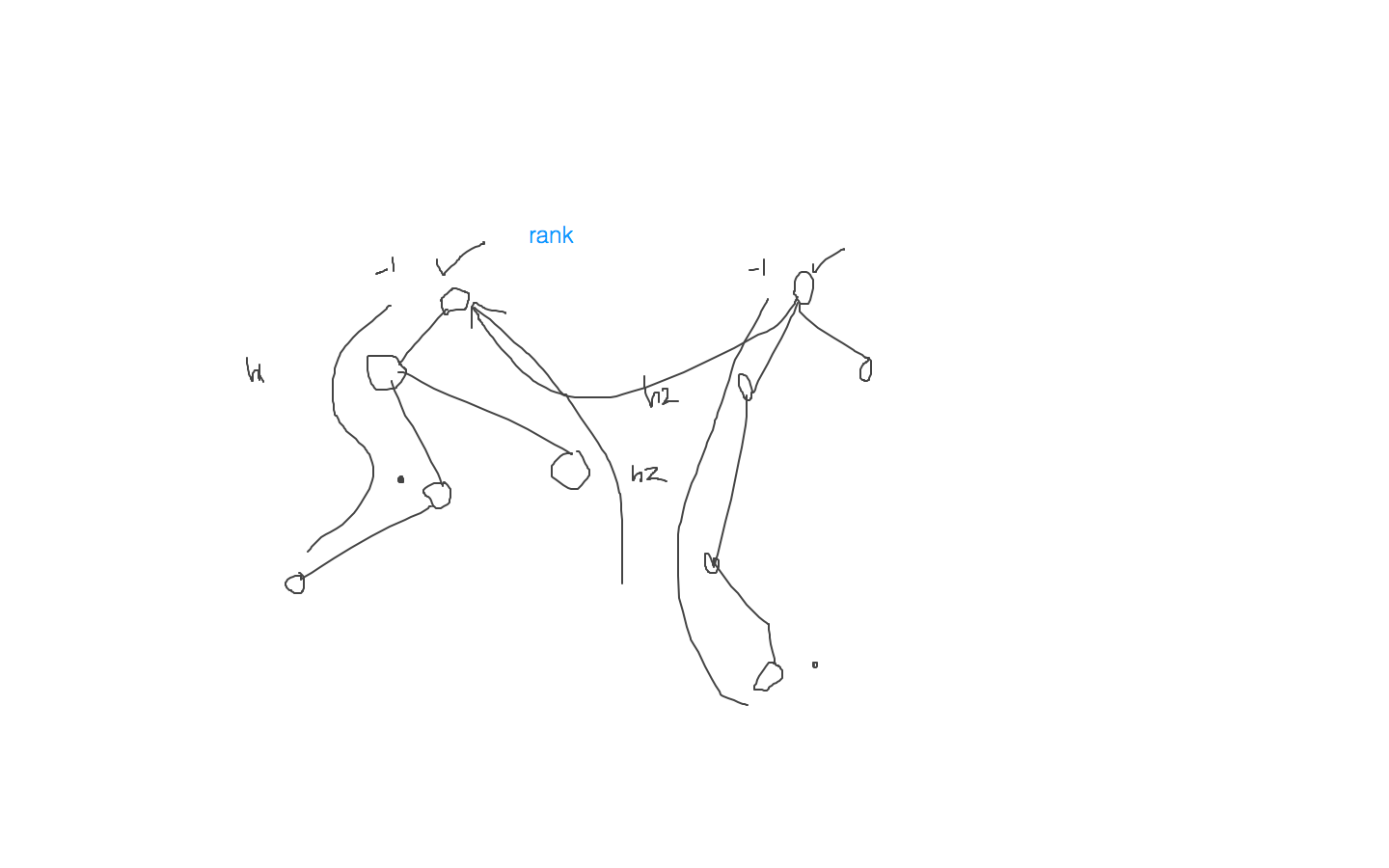
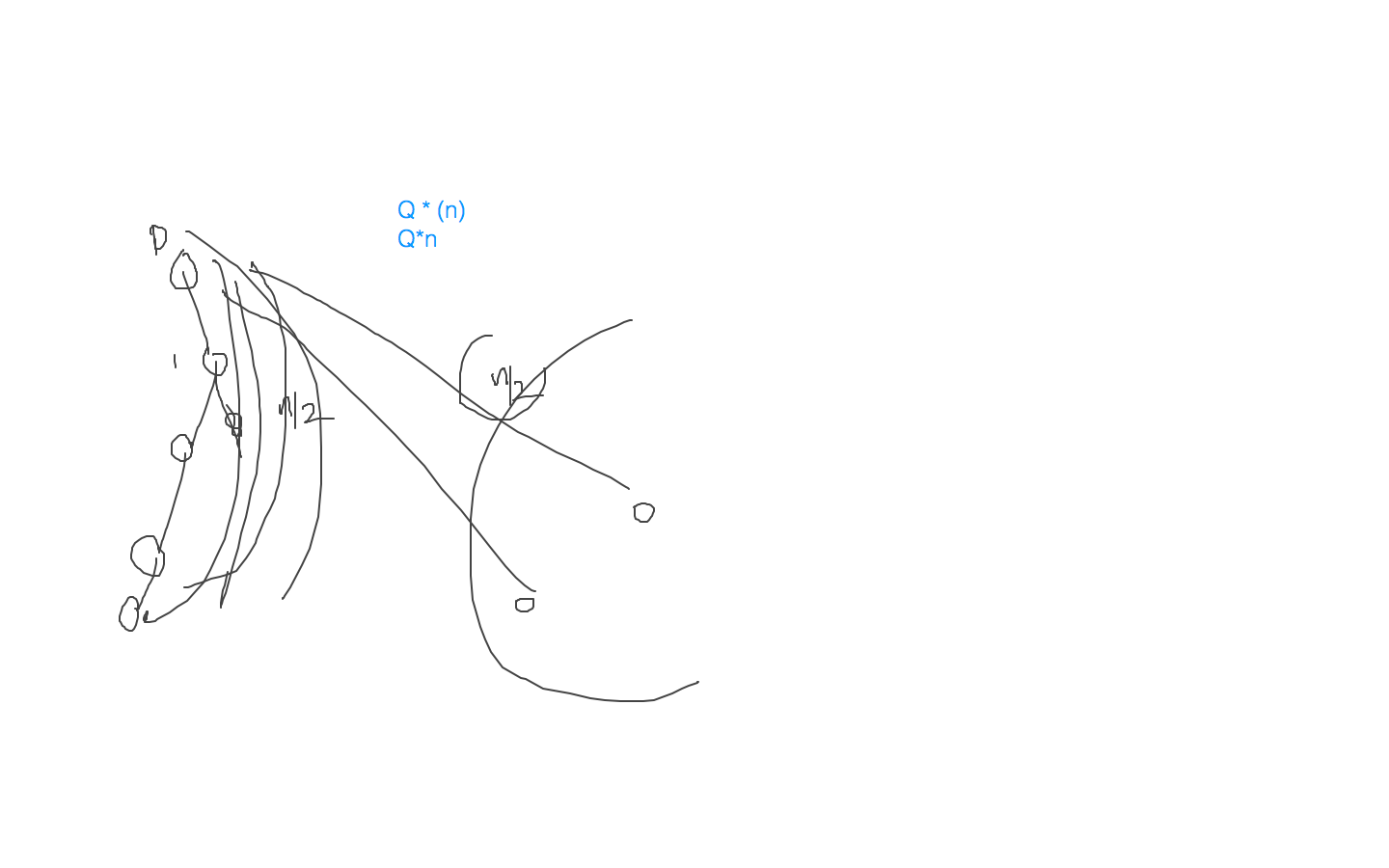
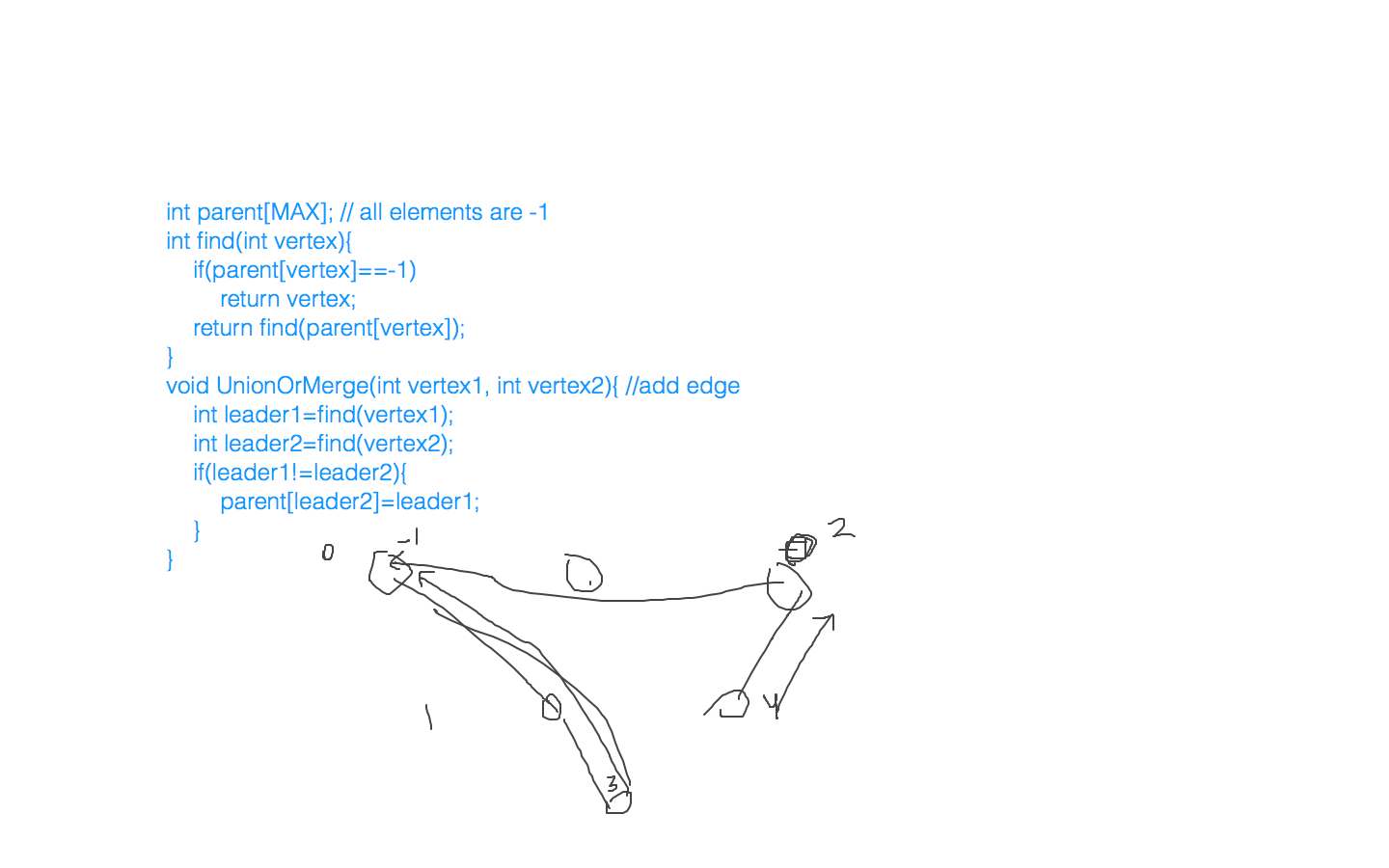
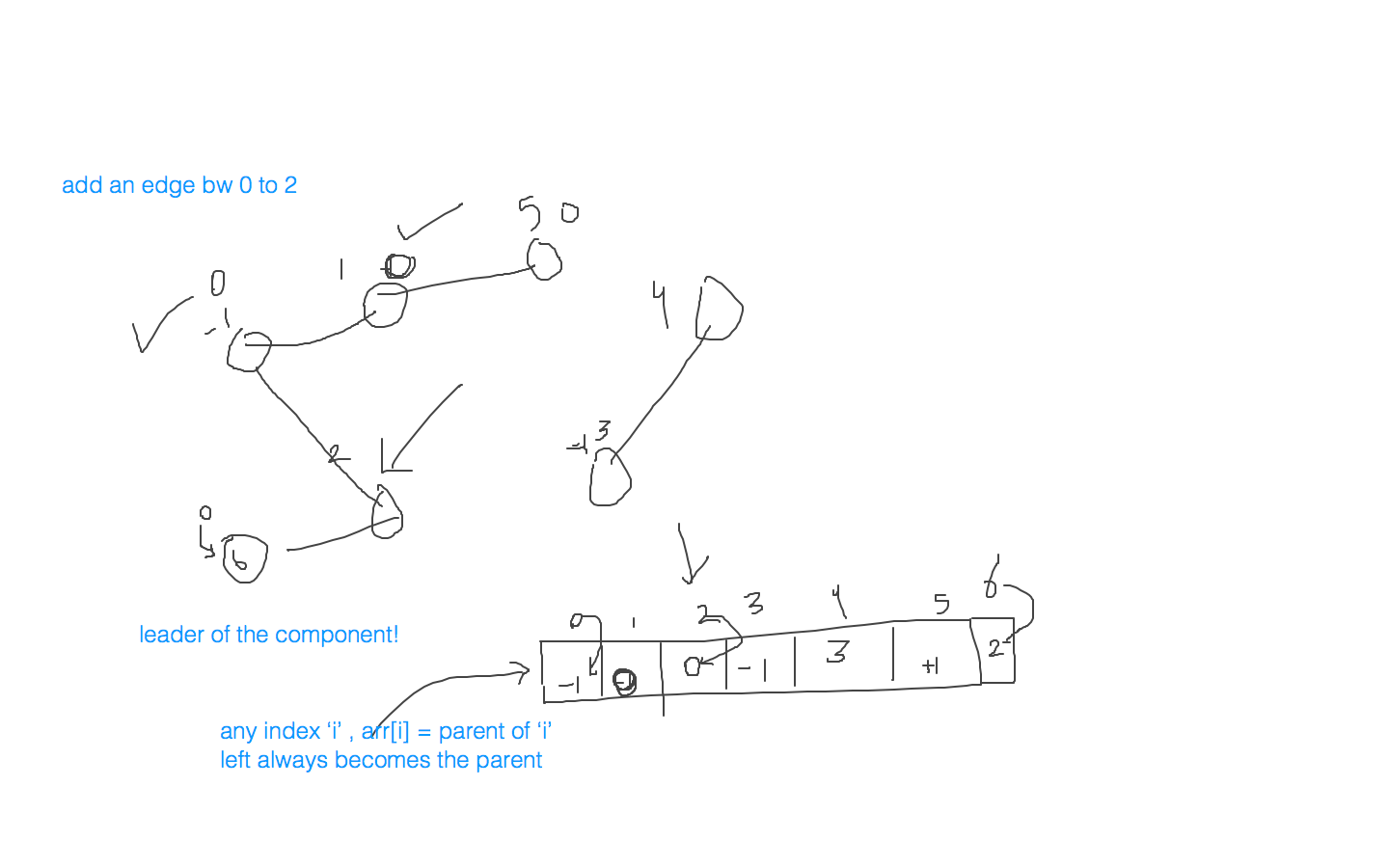
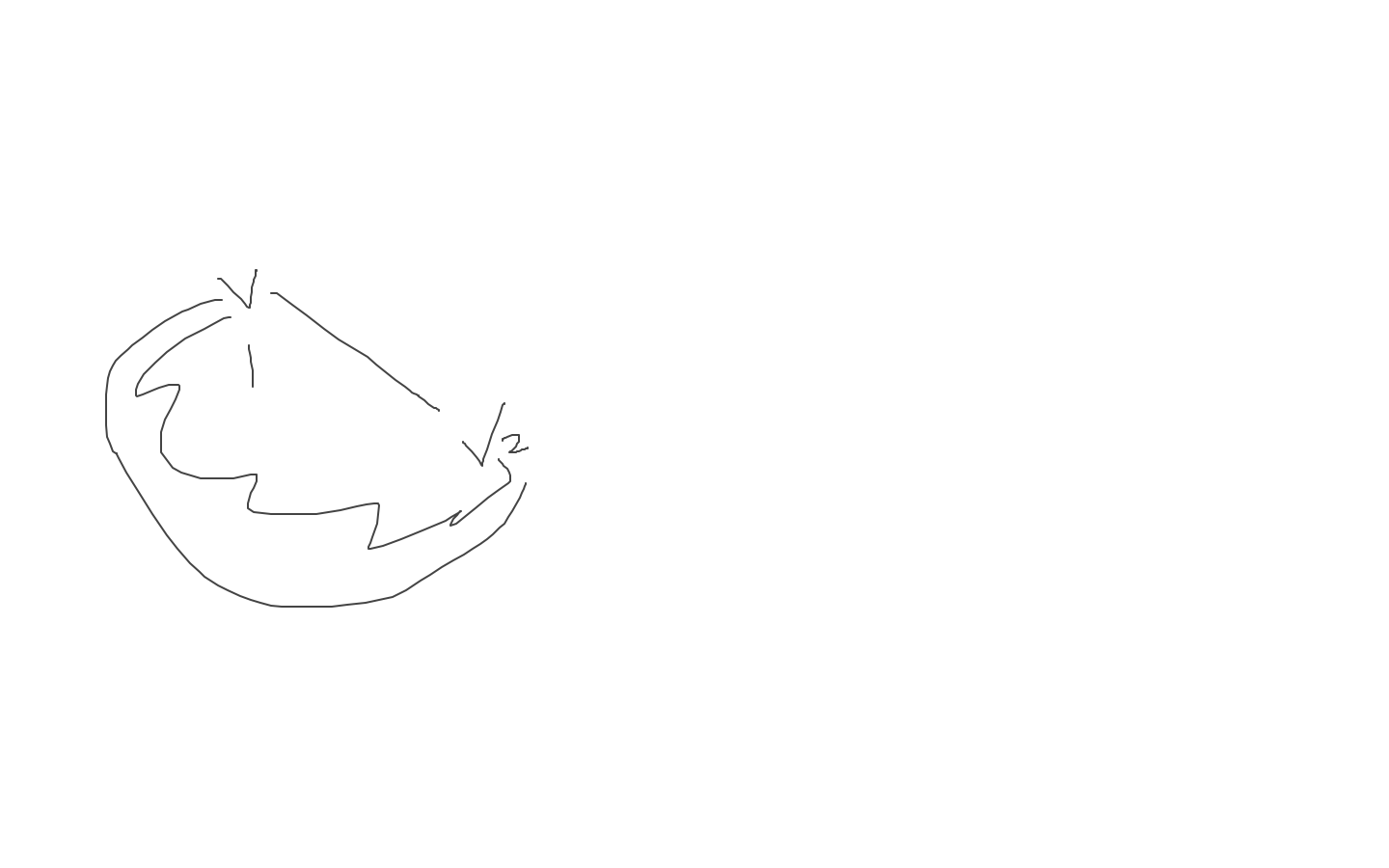
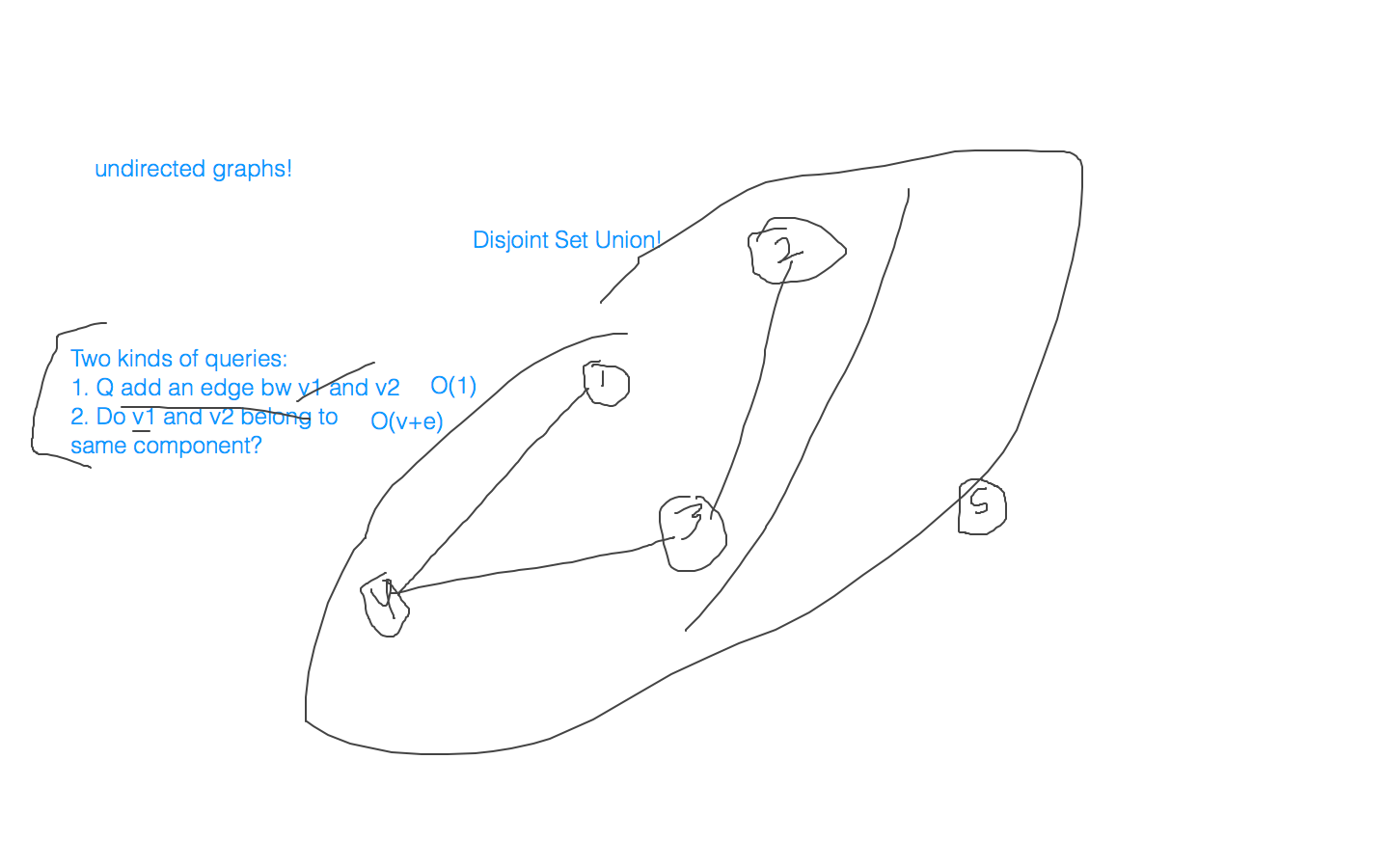
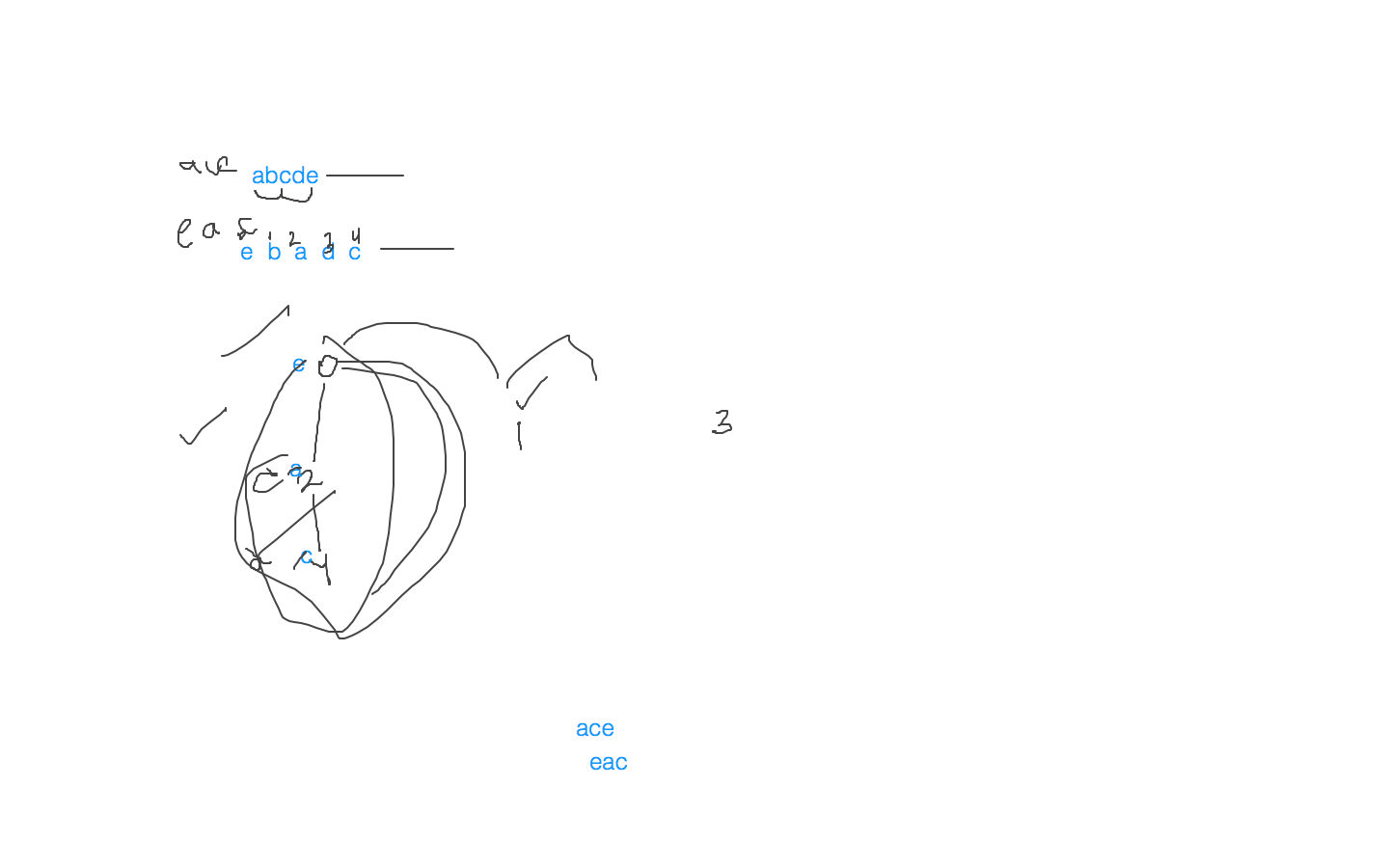
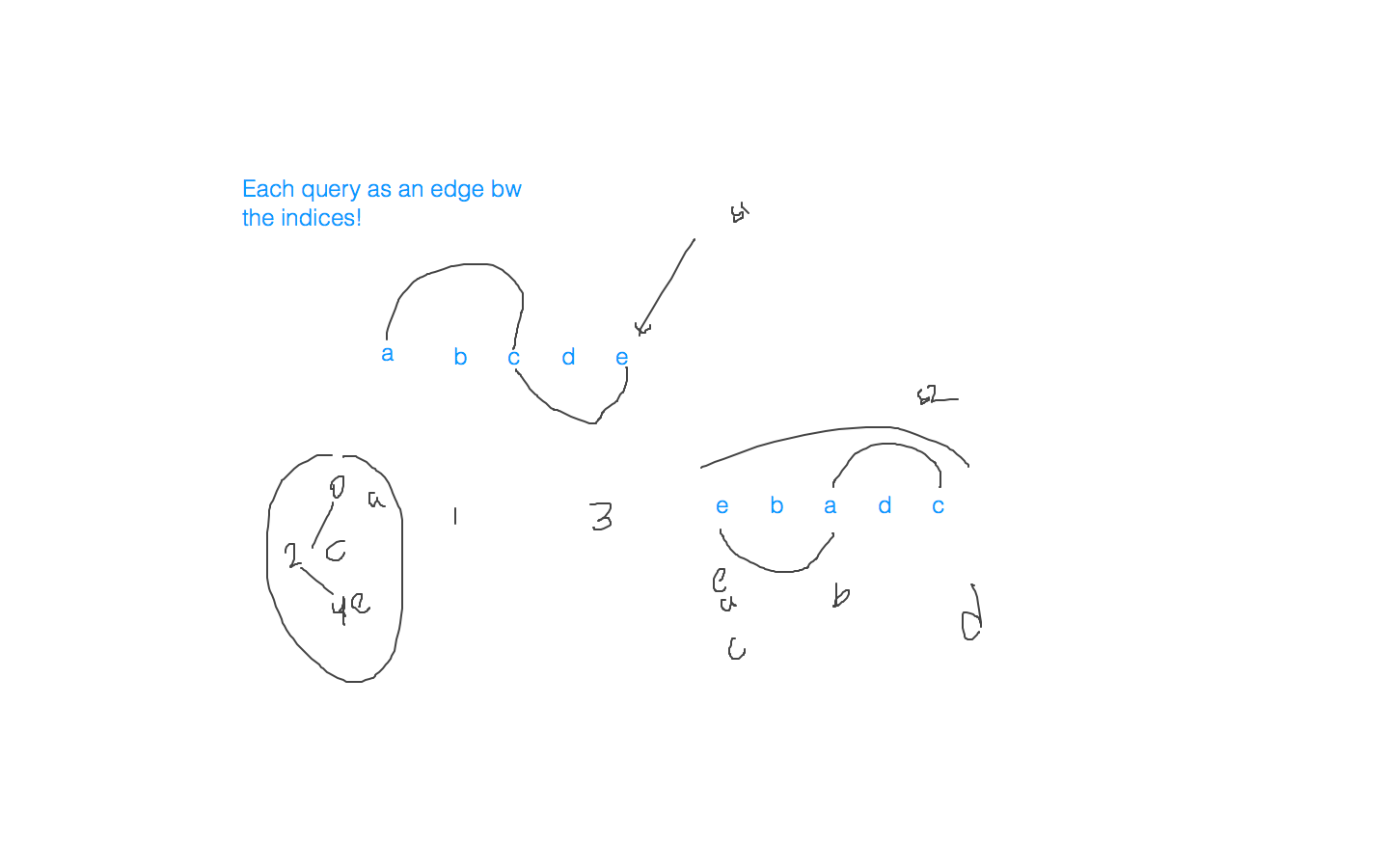
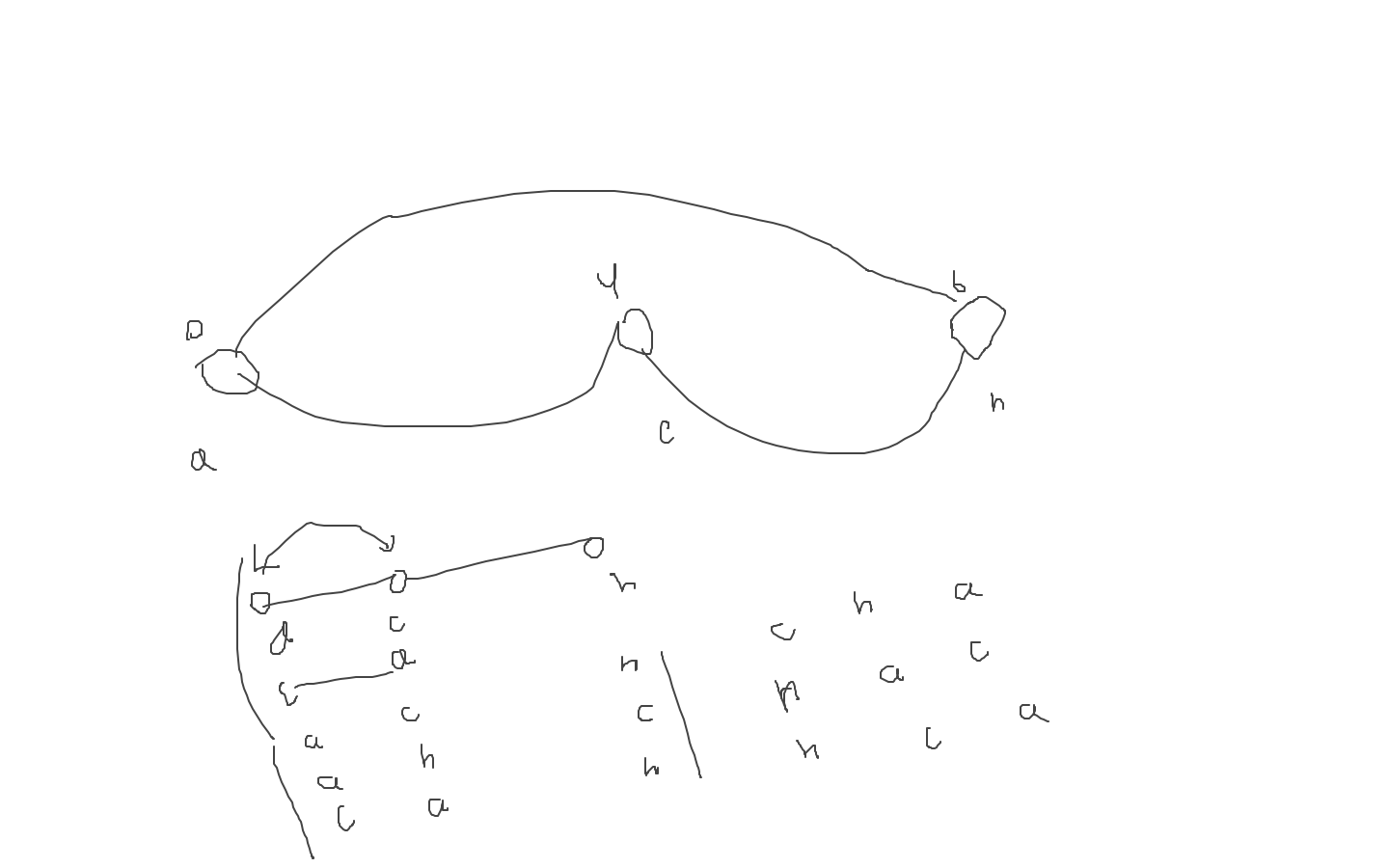
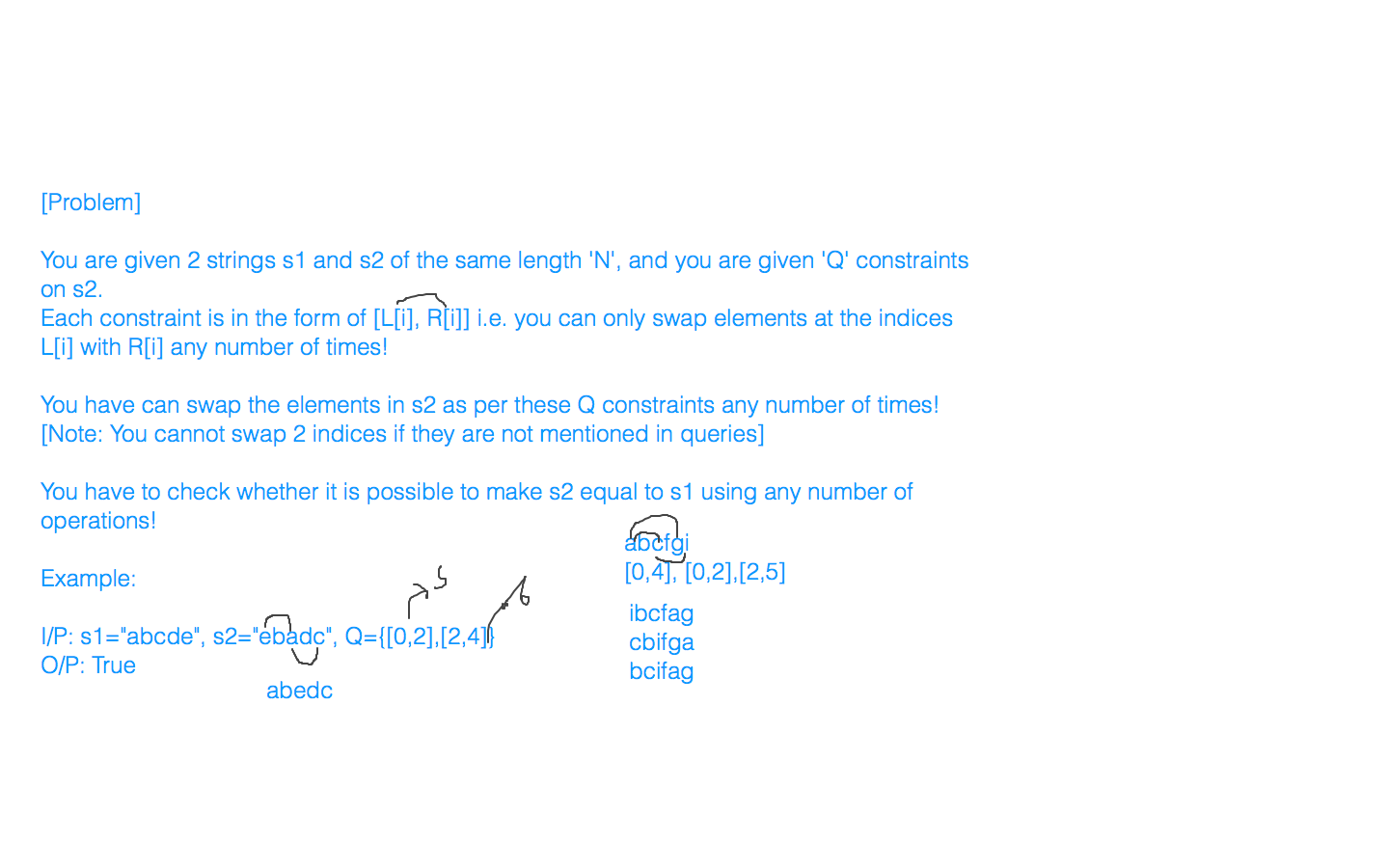
1. <https://www.slideshare.net/WeiLi73/time-complexity-of-union-find-55858534>

2. [https://en.wikipedia.org/wiki/Proof\_of\_O(log\*n)\_time\_complexity\_of\_union%E2%80%93find](https://en.wikipedia.org/wiki/Proof_of_O(log*n)_time_complexity_of_union%E2%80%93find)

[Visalization Tool]

<https://visualgo.net/en/ufds?slide=1>

|  |
| --- |
| //optmized int rank[MAX];//initially 0 int parent[MAX];// parent[i]=i int find(int vertex){  if(parent[vertex]!=vertex)//not leader  parent[vertex]=find(parent[vertex]);//path compression  return (parent[vertex]); }  void Union(int vertex1, int vertex2){  int leader1=find(vertex1);  int leader2=find(vertex2);  //Union by rank  if(rank[leader1]>rank[leader2]){  parent[leader2]=leader1;  }  else if(rank[leader1]<rank[leader2]){  parent[leader1]=leader2;  }  else{  parent[leader2]=leader1;  rank[leader1]++;  } } // Not optmized int find(int vertex){  if(parent[vertex]==-1)//not leader  return vertex;  return find(parent[vertex]); } void Union(int vertex1, int vertex2){  int leader1=find(vertex1);  int leader2=find(vertex2);  if(leader1!=leader2)  parent[leader2]=leader1; } |



**[14th July 2020, 8:30-11:00pm(IST)](Sachin)**

**[Class: Graphs, MST, Dijkstra’s]**

Links:

→ <https://www.geeksforgeeks.org/union-find-algorithm-set-2-union-by-rank/>  
→ <https://www.geeksforgeeks.org/find-the-number-of-islands-set-2-using-disjoint-set/>

→ <https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/>

→ <http://cseweb.ucsd.edu/~kube/cls/100/Lectures/lec14.spanning/lec14-11.html>

→ <https://www.geeksforgeeks.org/bipartite-graph/>

→ <https://www.geeksforgeeks.org/minimum-number-of-colors-required-to-color-a-graph/>

→ <https://www.geeksforgeeks.org/shortest-path-unweighted-graph/>

→ <https://www.geeksforgeeks.org/shortest-path-in-a-directed-graph-by-dijkstras-algorithm/?ref=rp>

→ <https://www.geeksforgeeks.org/dijkstras-algorithm-for-adjacency-list-representation-greedy-algo-8/>

→ <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-using-priority_queue-stl/>

→ <https://www.geeksforgeeks.org/applications-of-minimum-spanning-tree/?ref=rp>

→ <https://stackoverflow.com/questions/4412031/what-are-the-applications-of-the-shortest-path-algorithm>

Extra:

→ <https://www.geeksforgeeks.org/interesting-shortest-path-questions-set-1/>

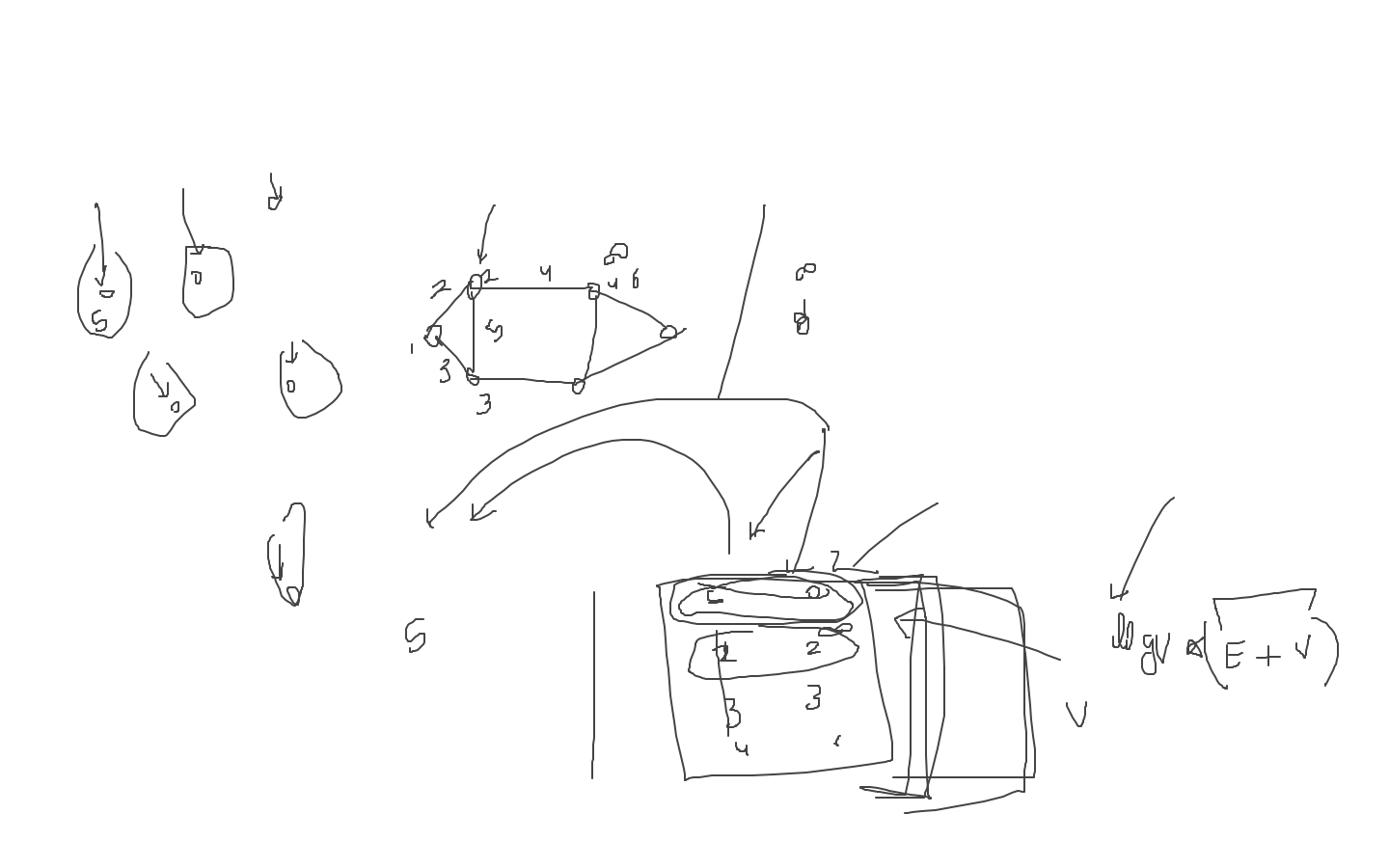
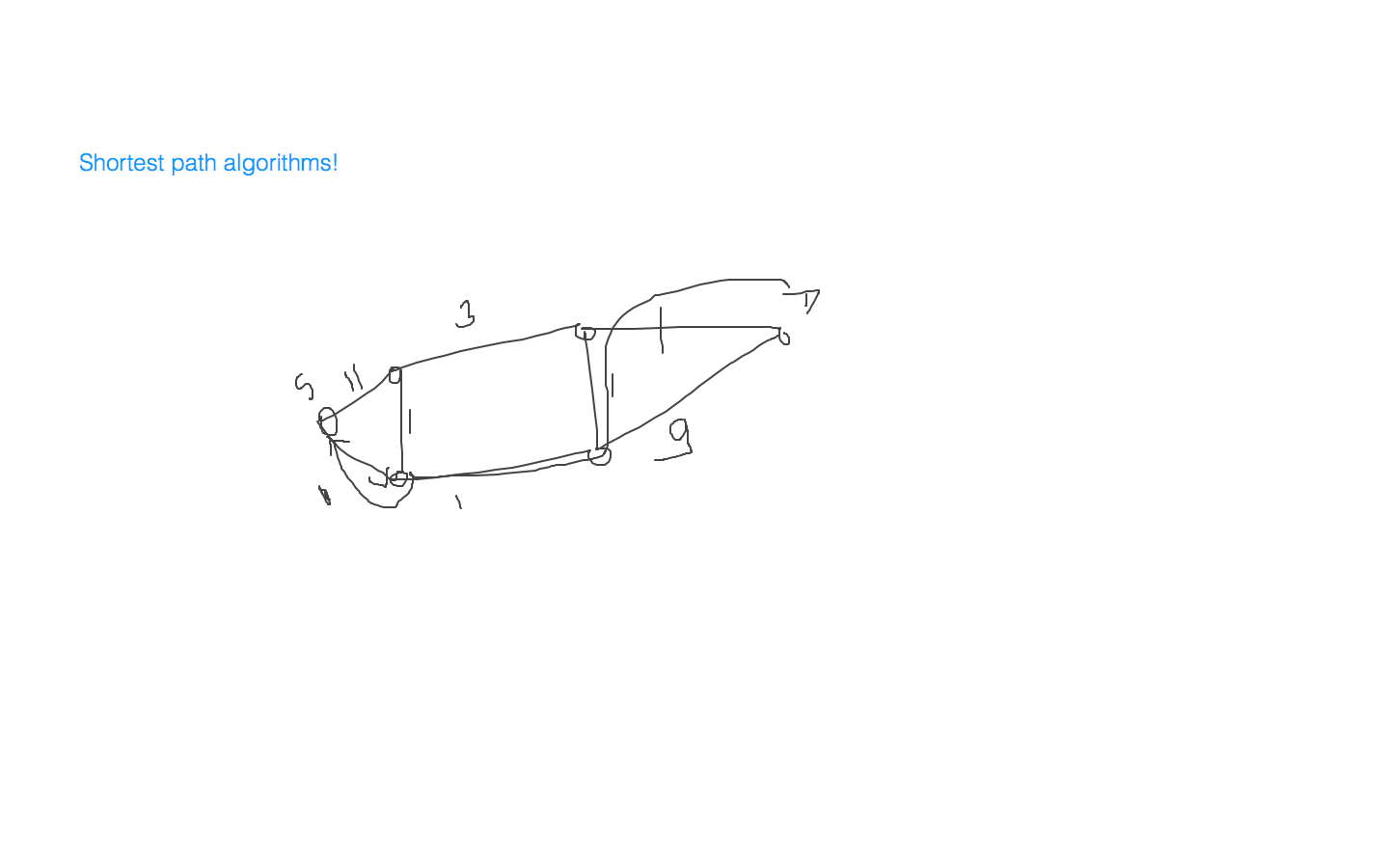
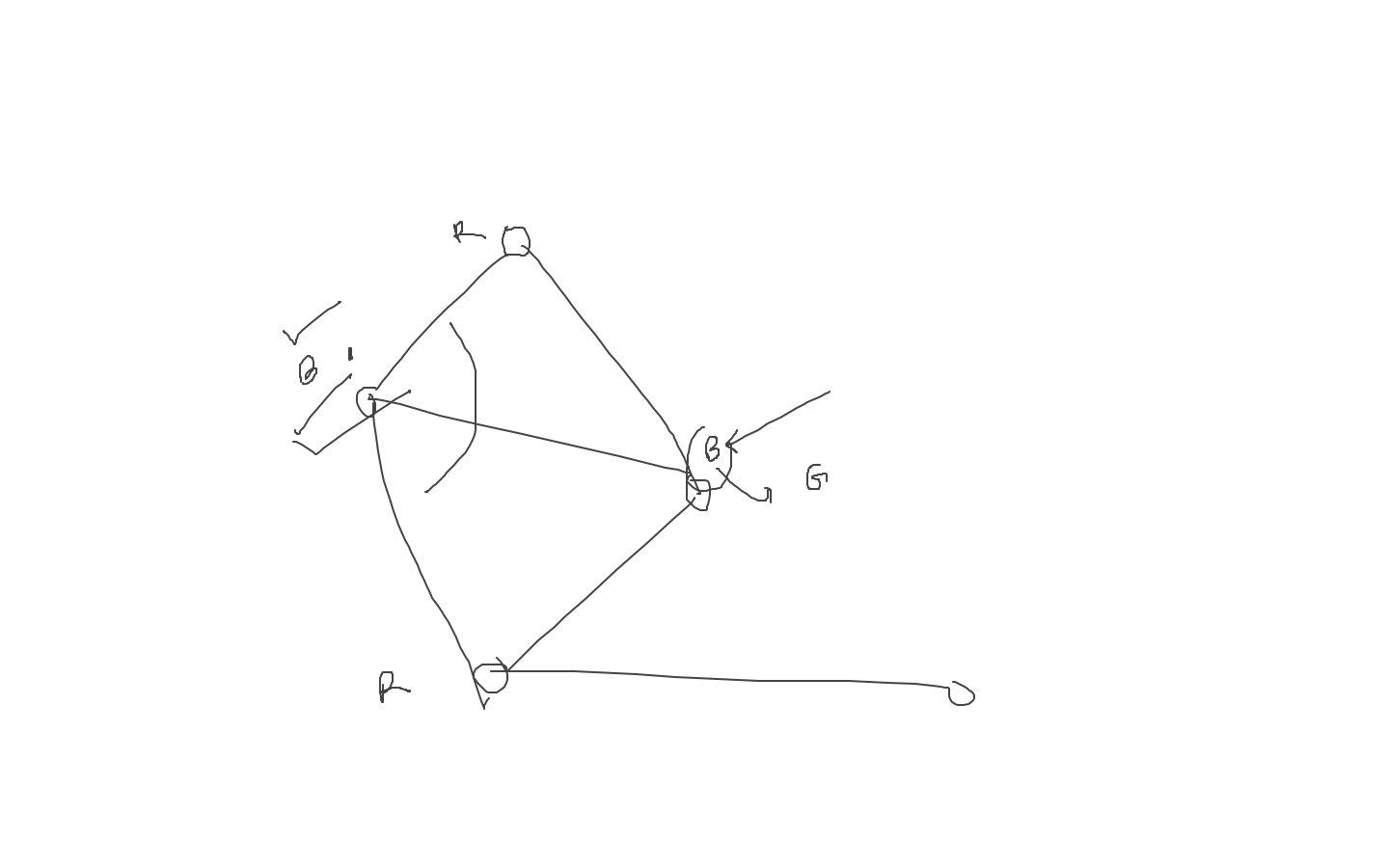
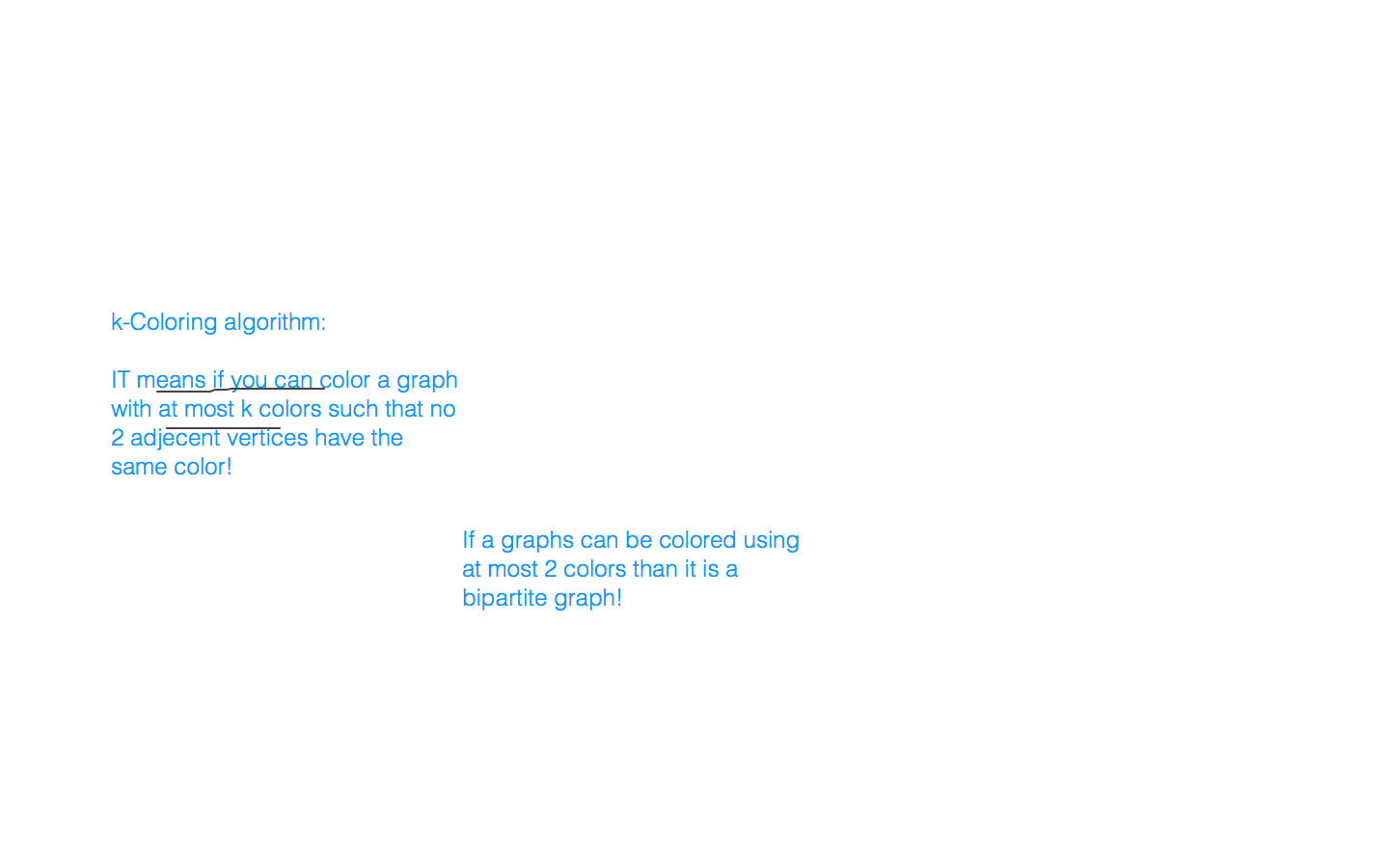
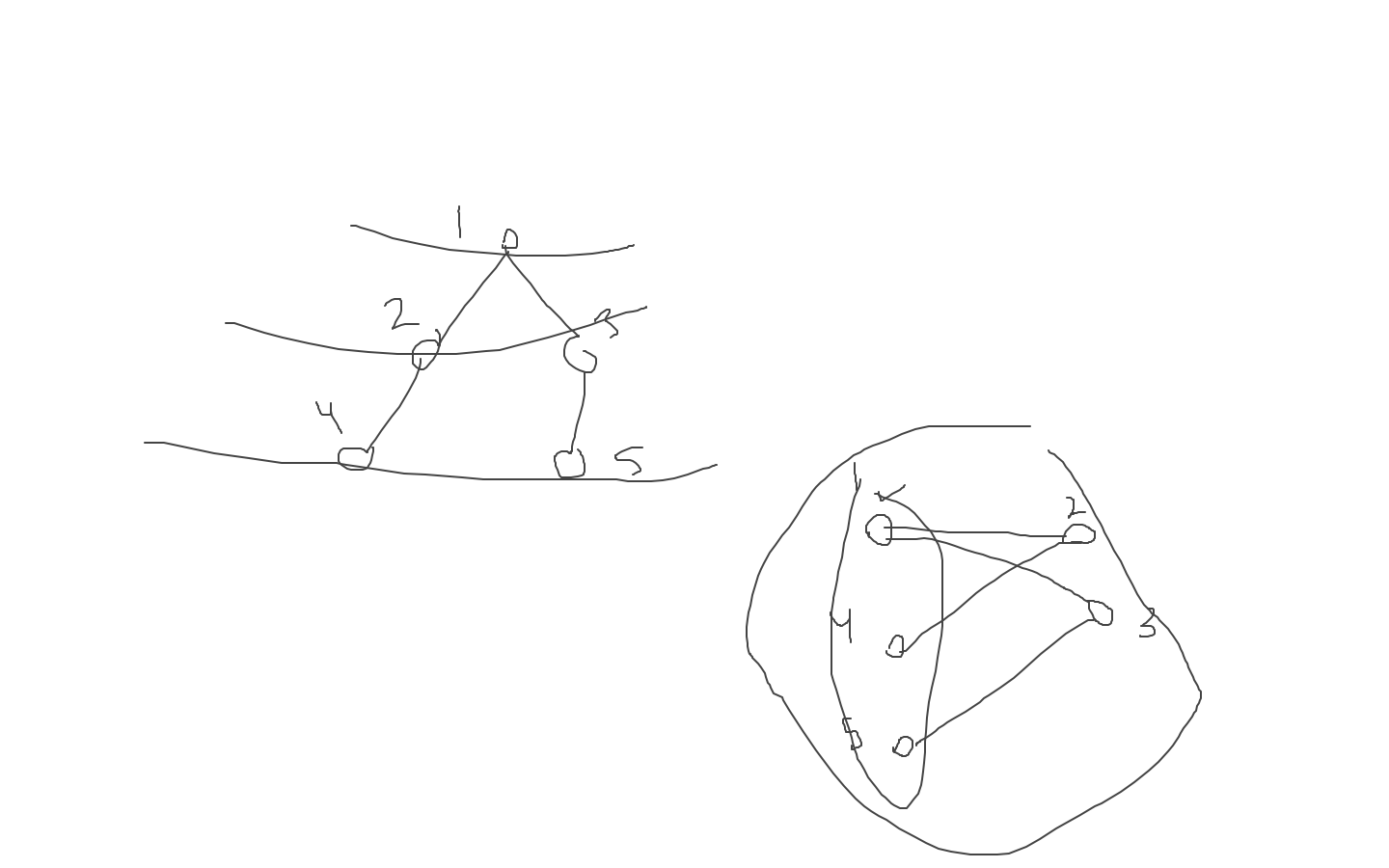
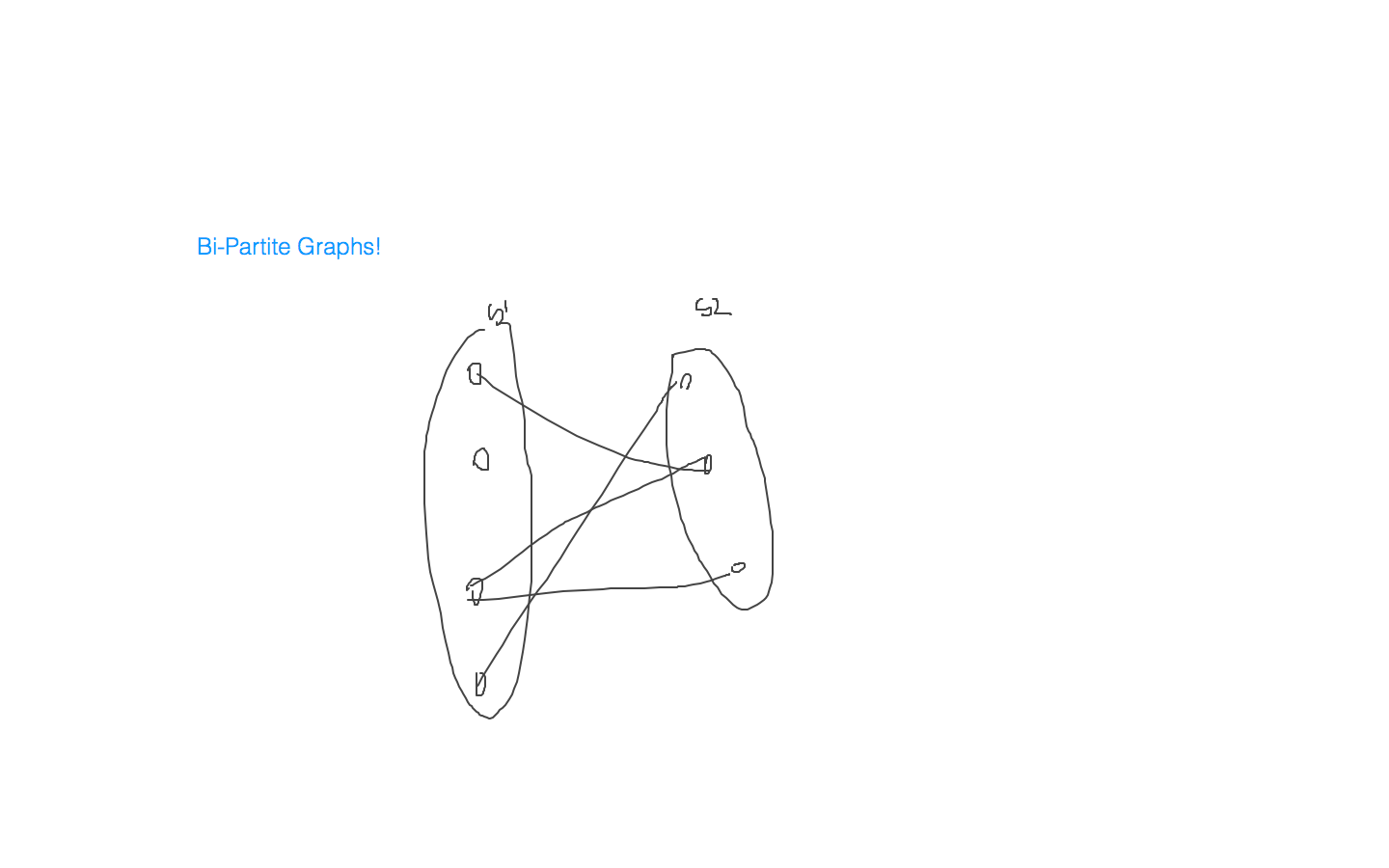
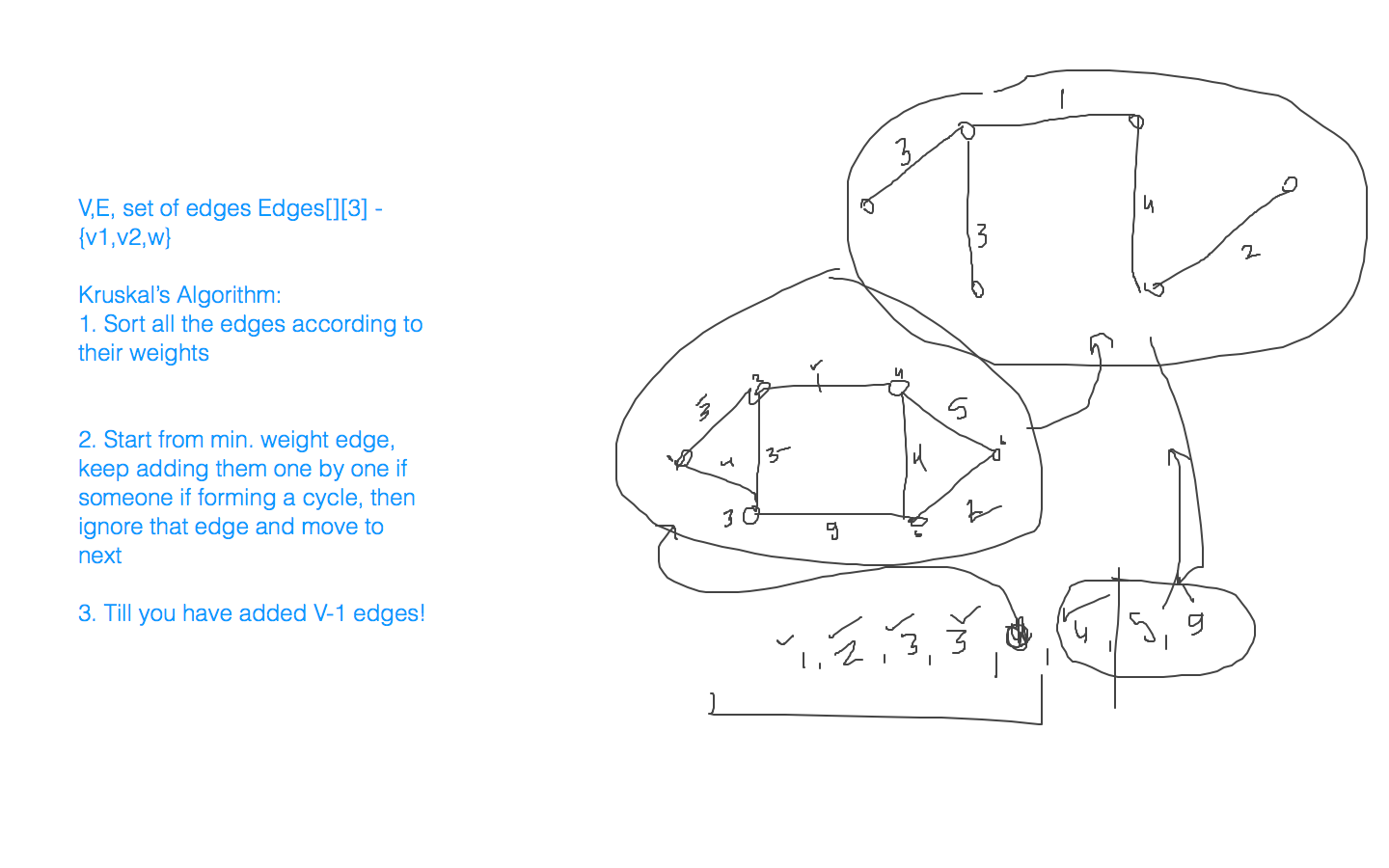
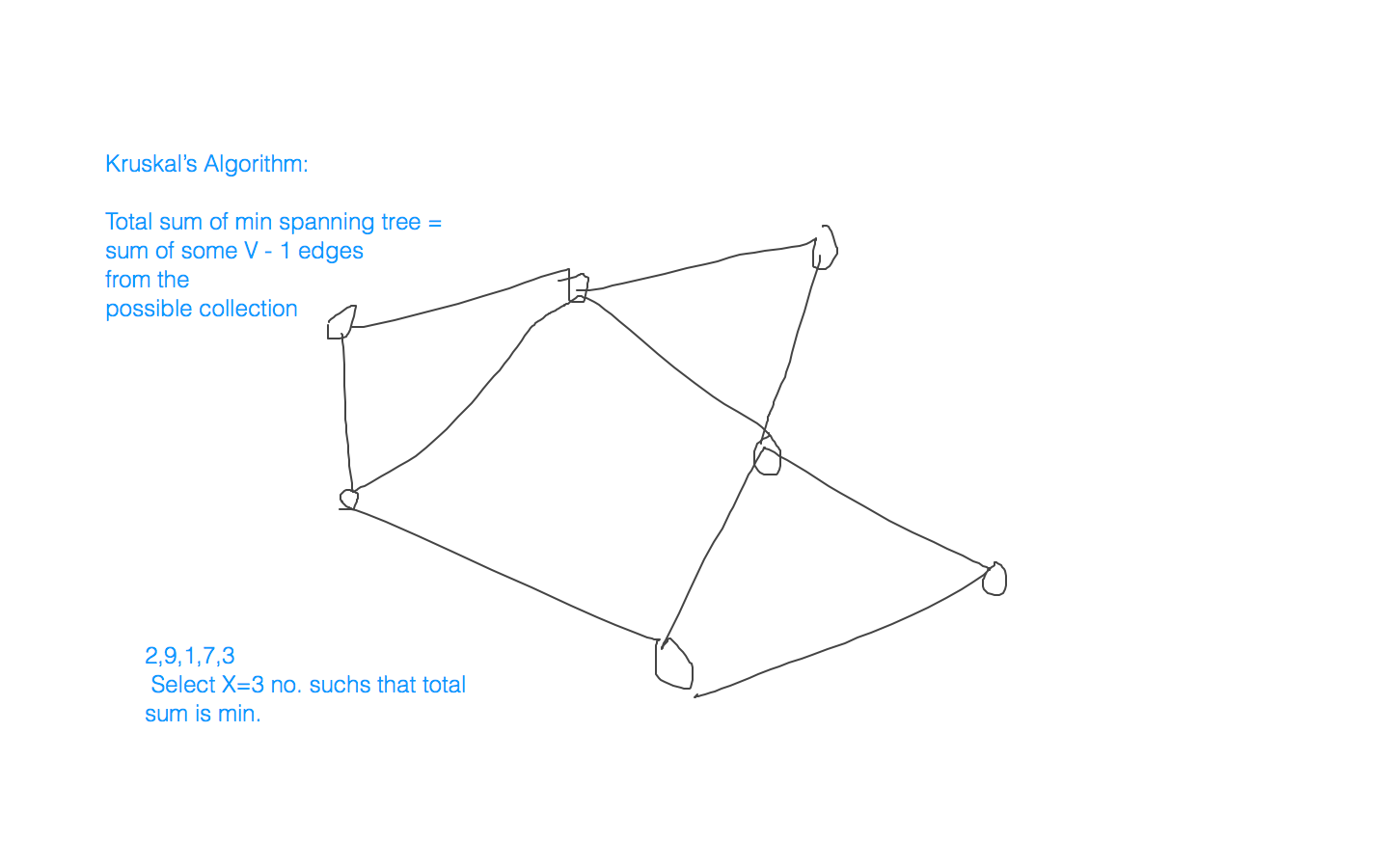
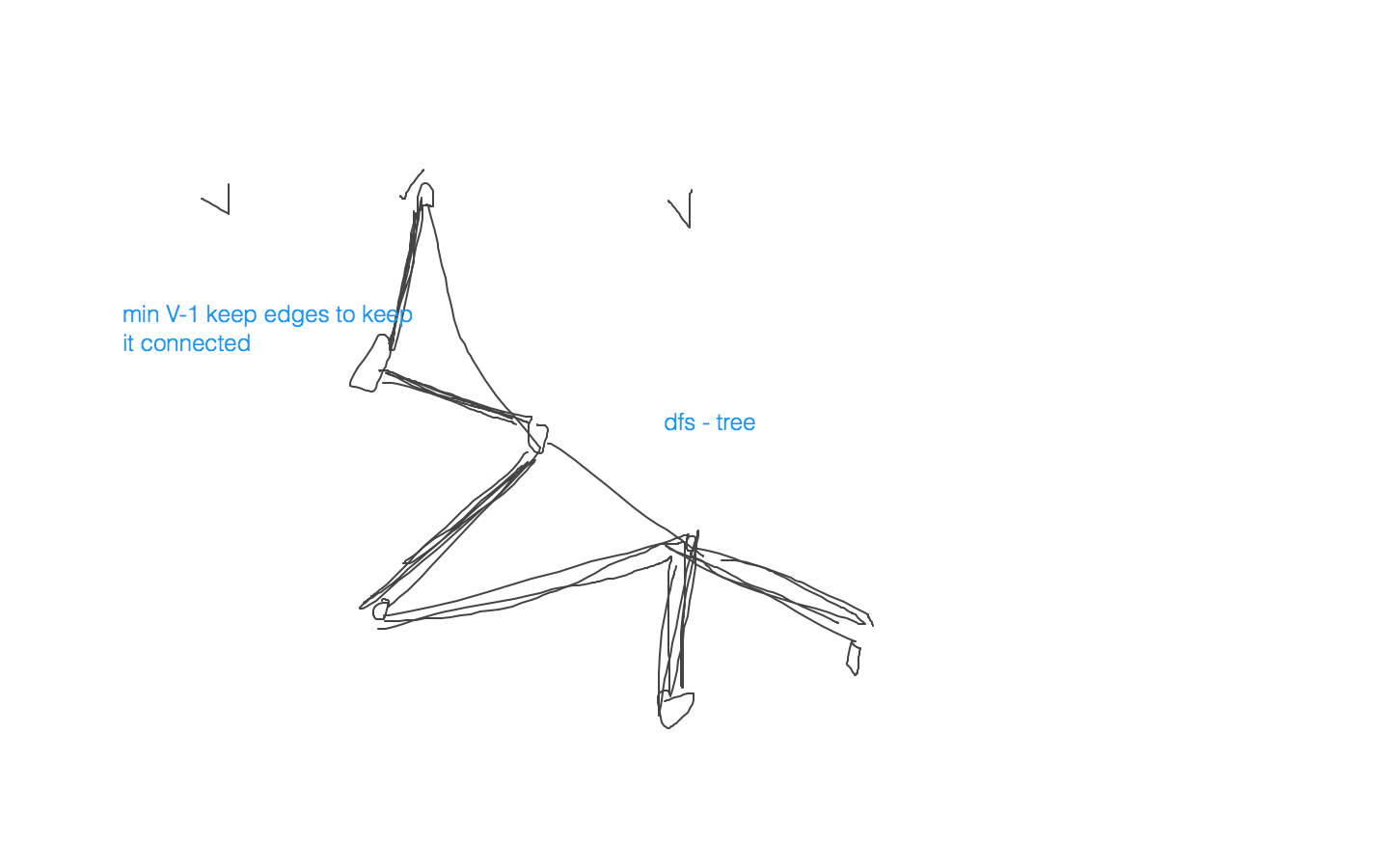
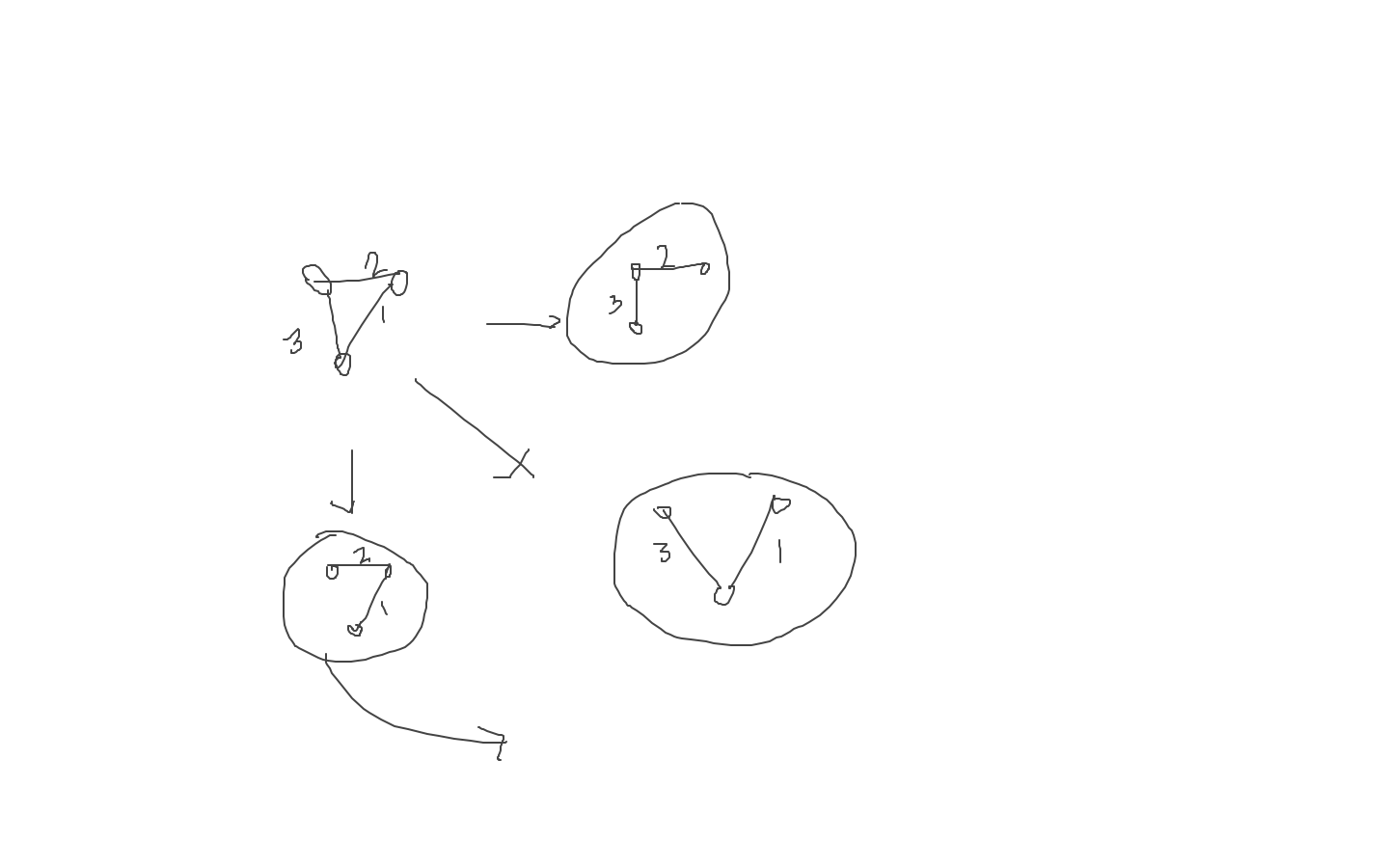
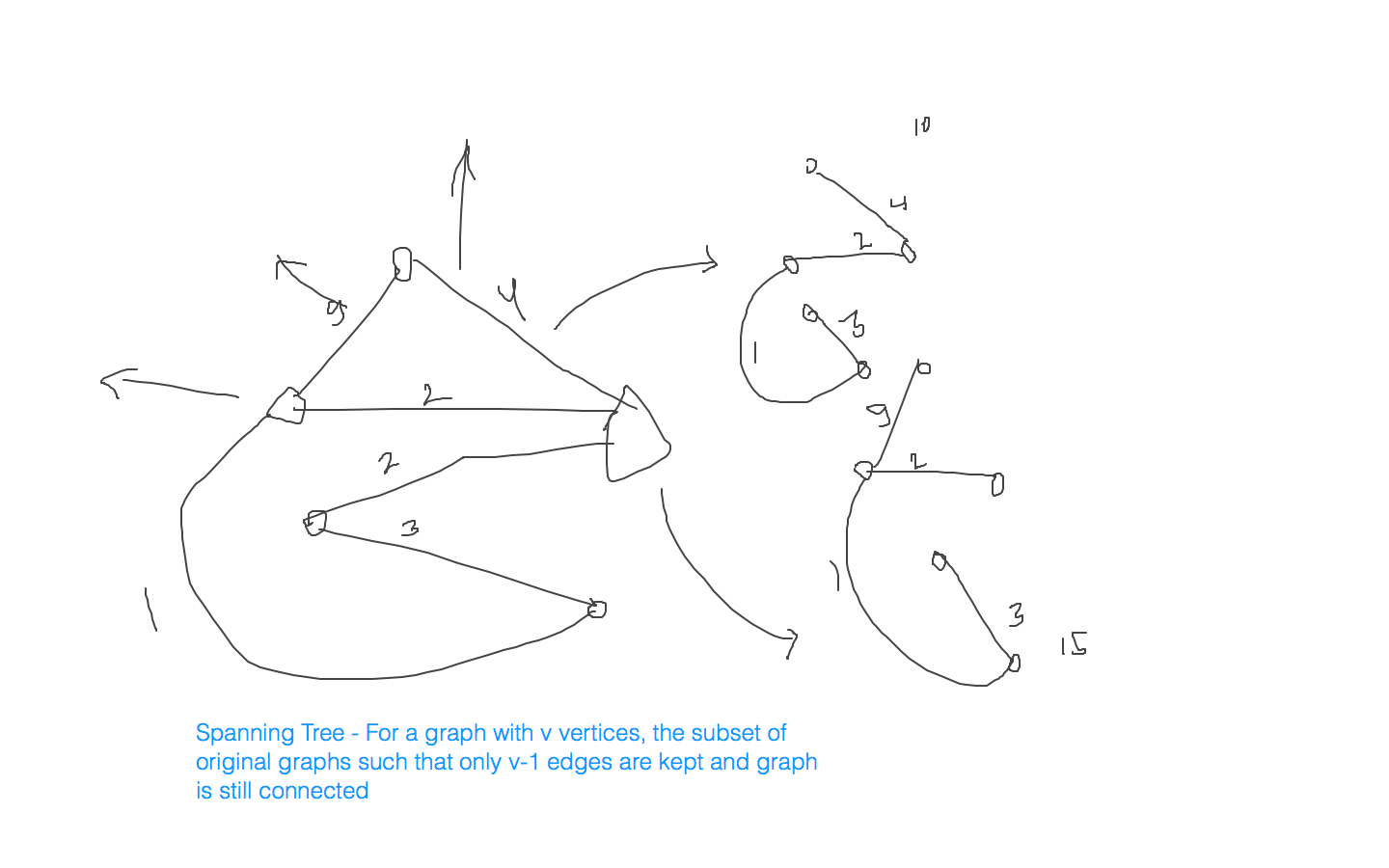
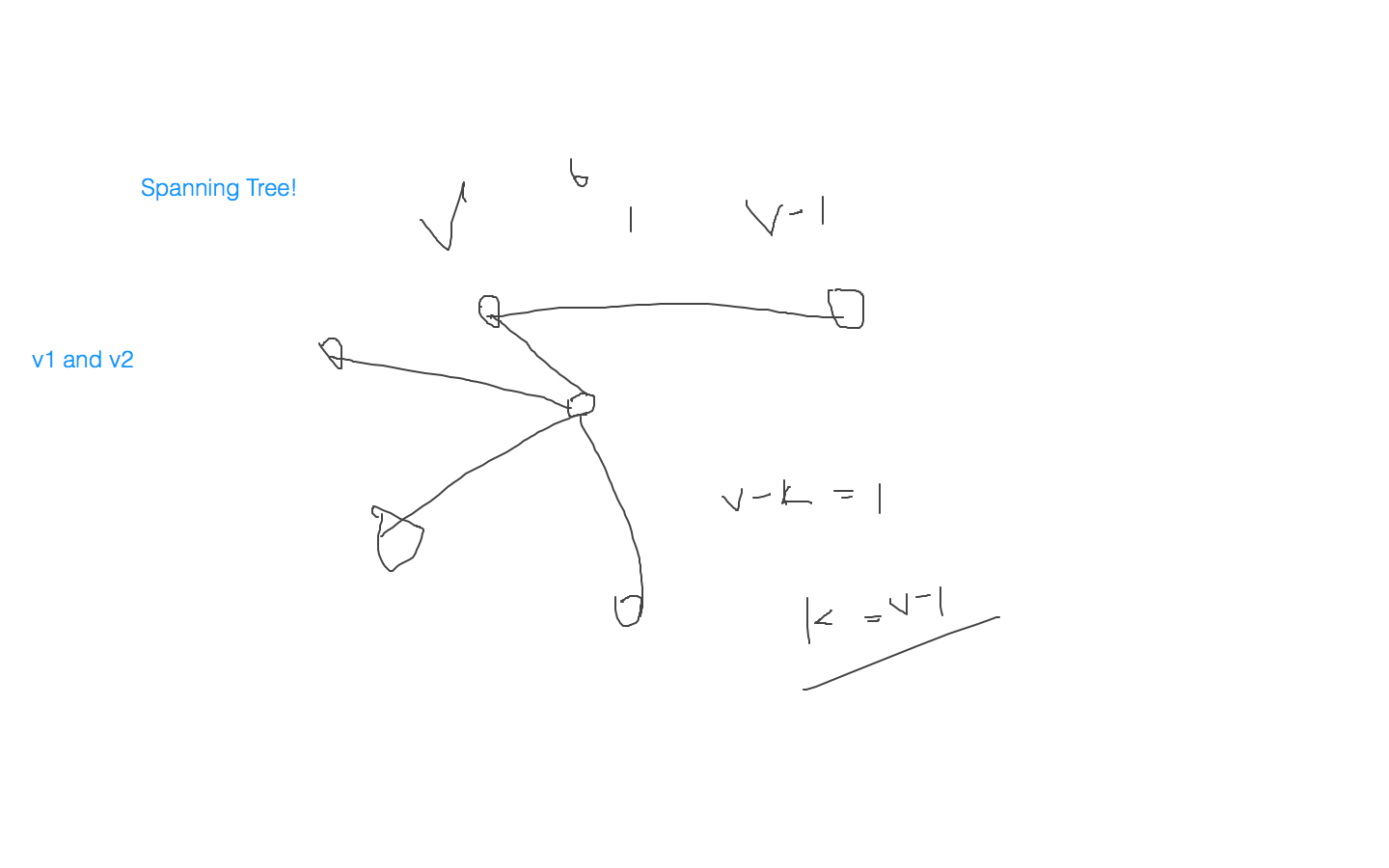
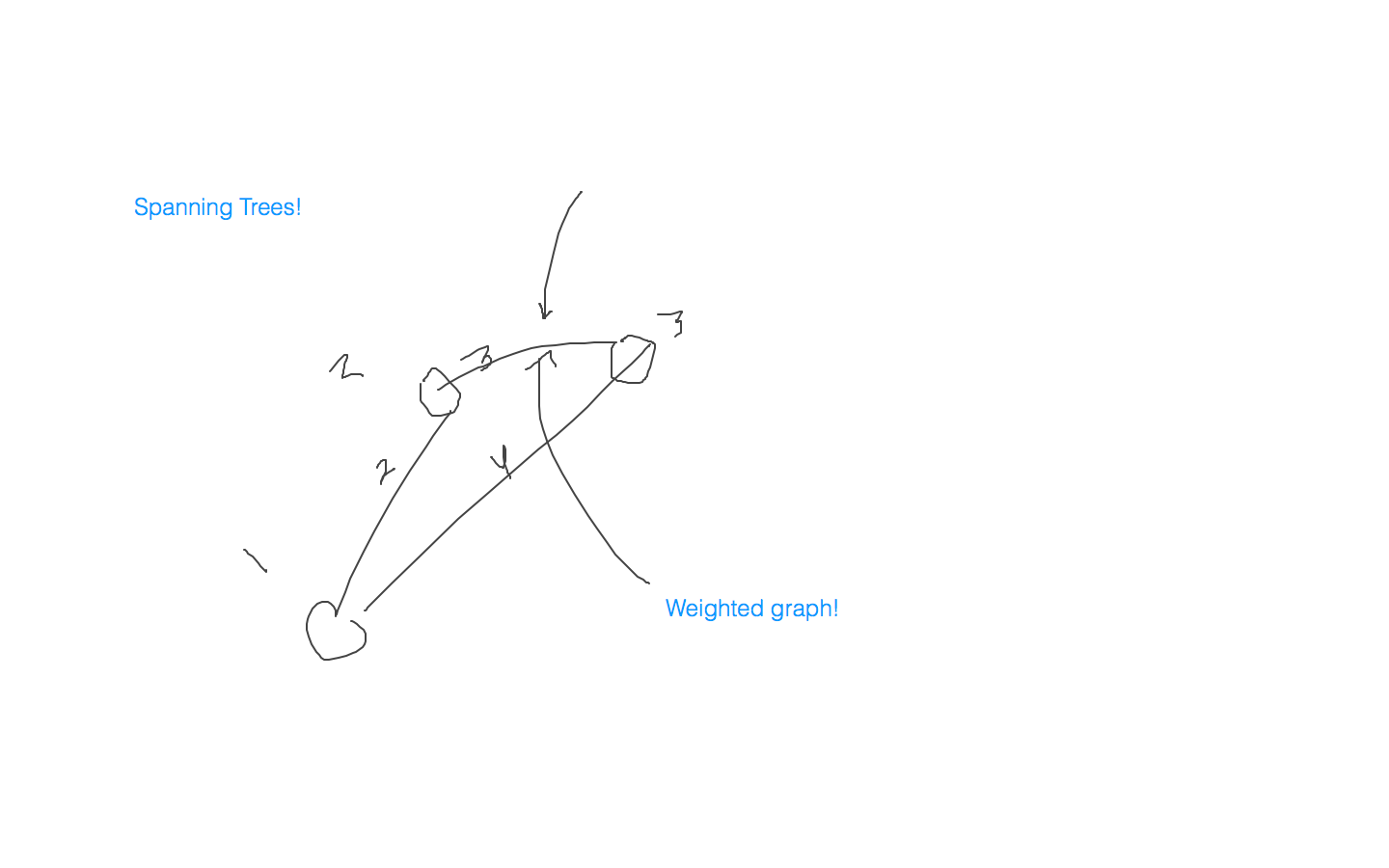
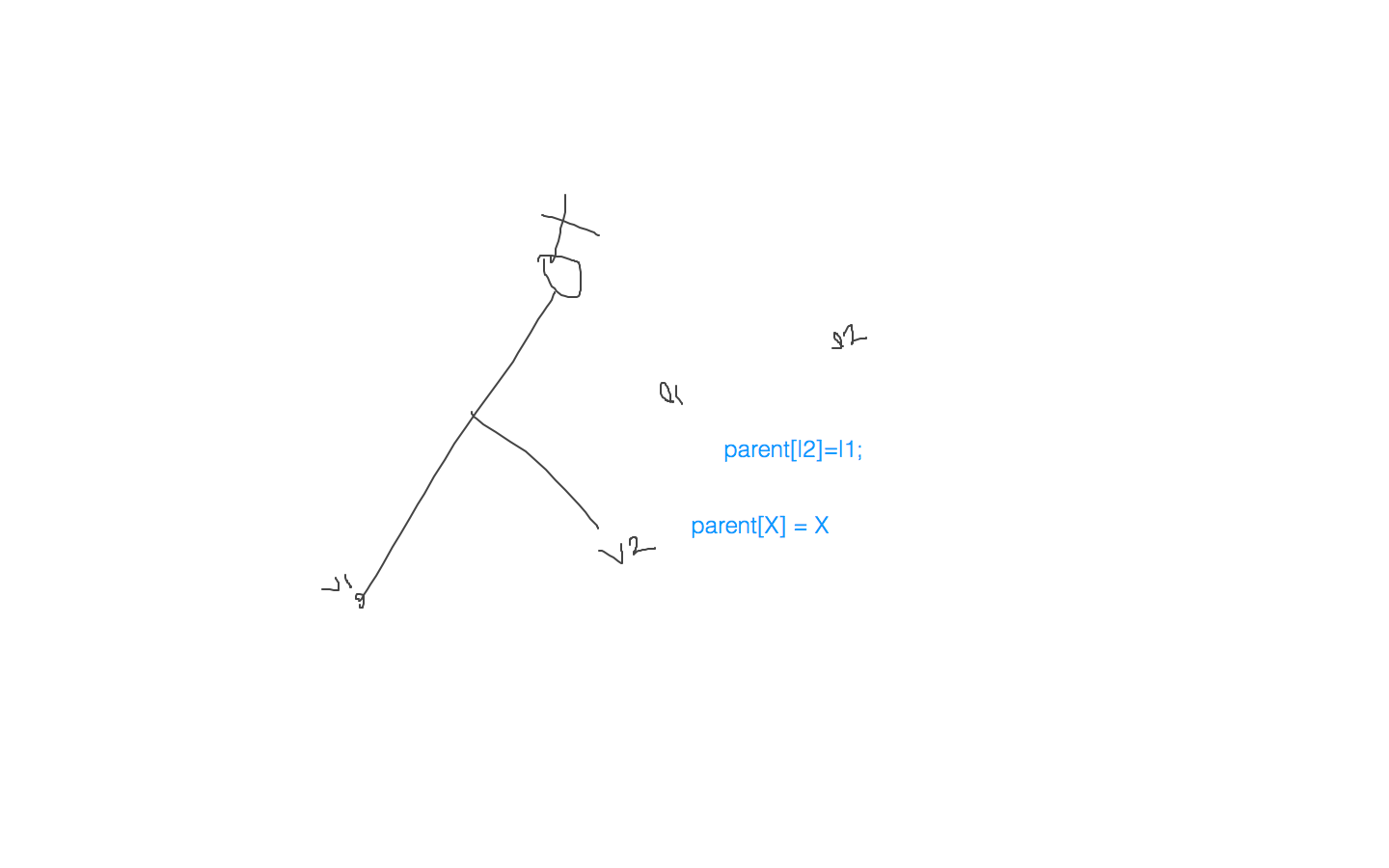
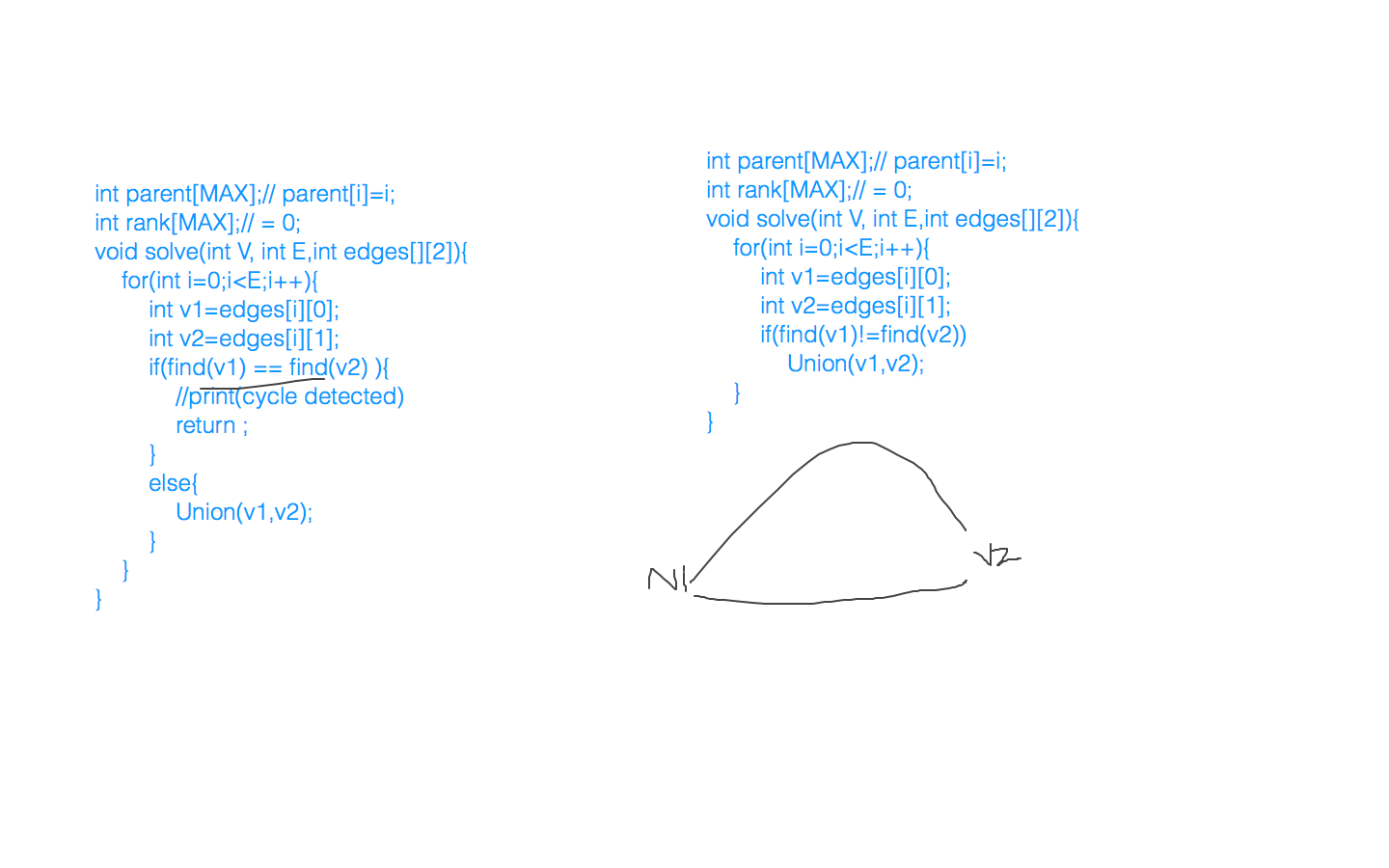
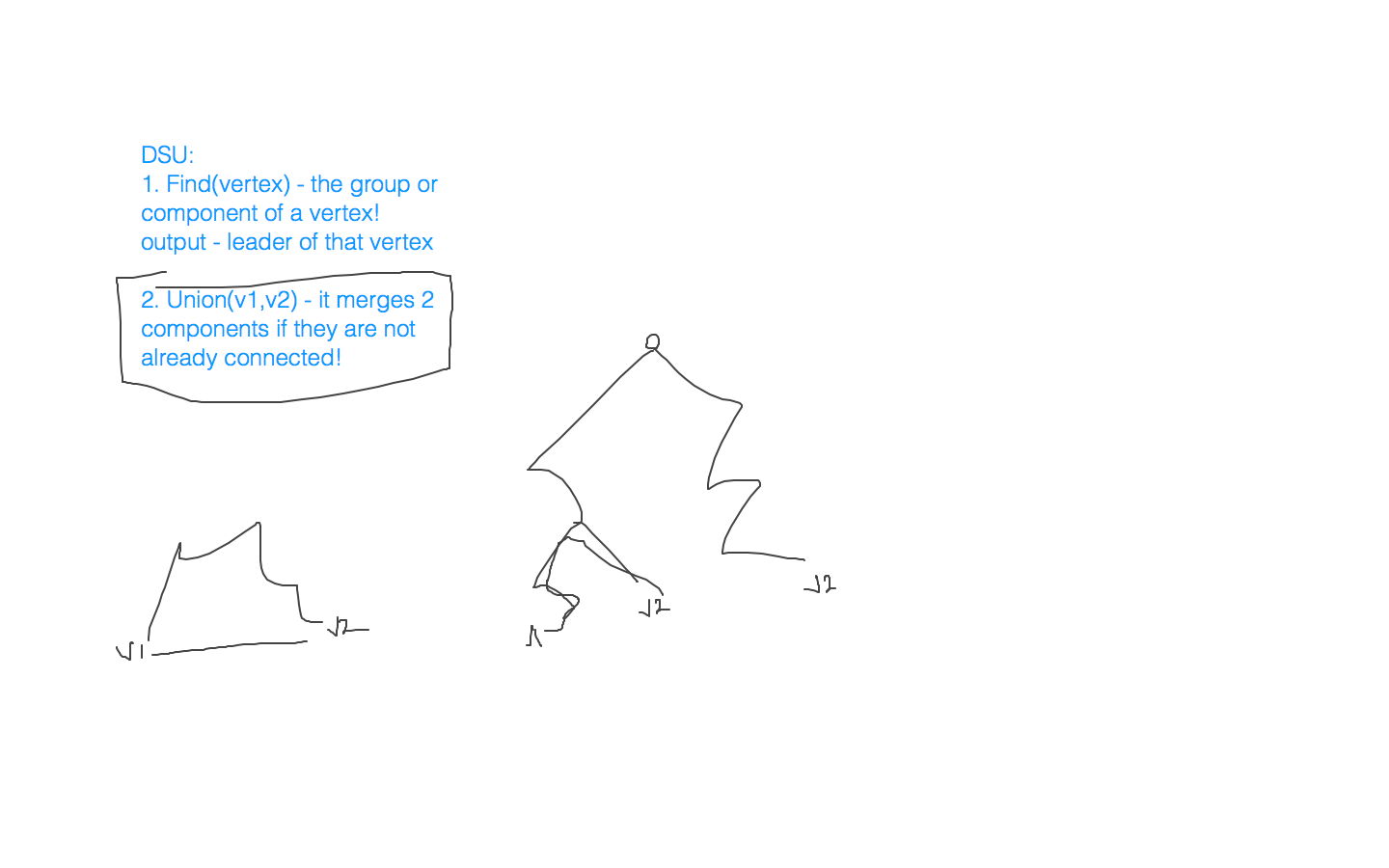
Visualization tool:

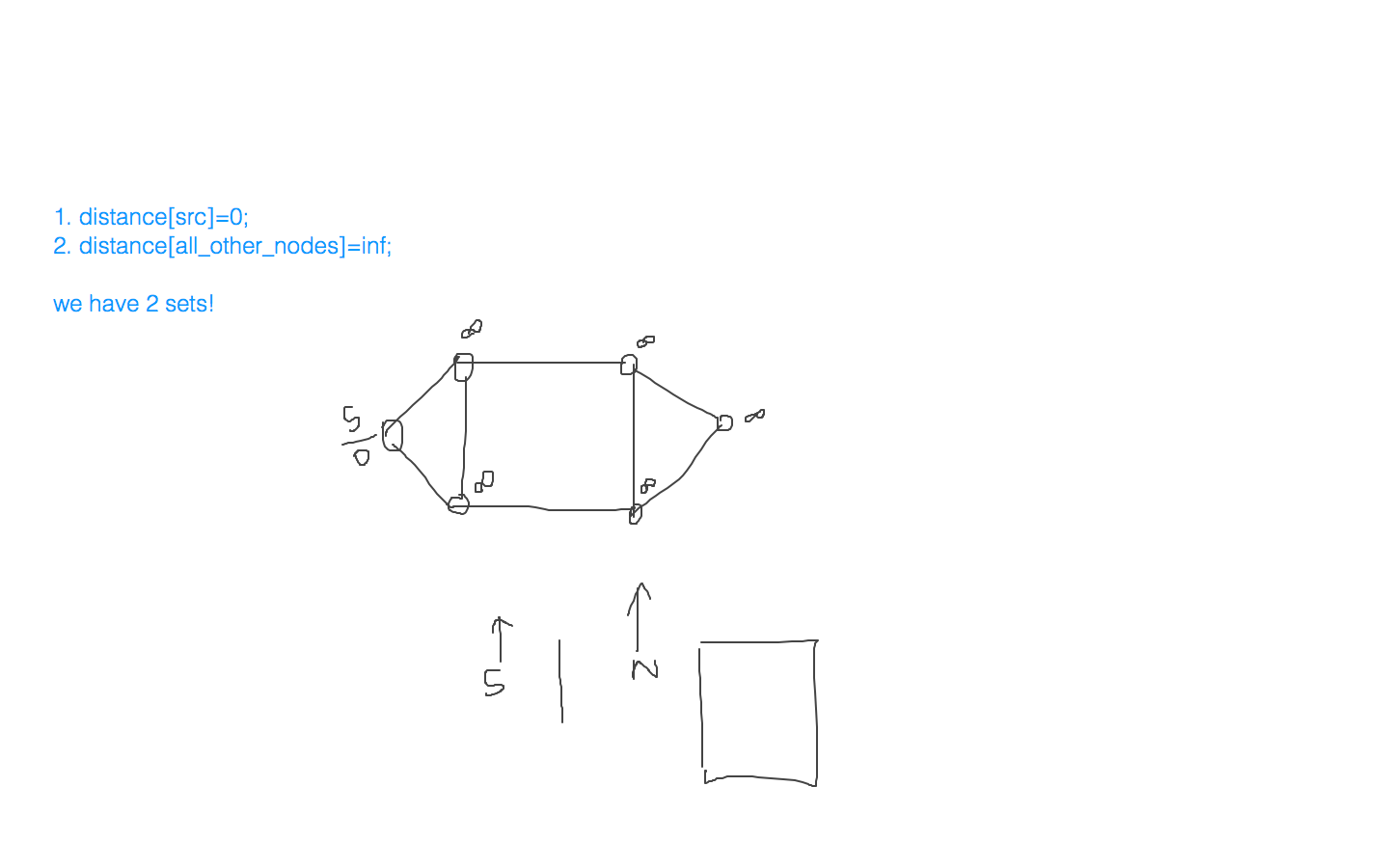
→ <https://visualgo.net/en/mst?slide=1>

→ <https://visualgo.net/en/sssp?slide=1>

Class notes:

|  |
| --- |
| /\*   \*/  int adj[MAX]; int color[MAX]; // -1 = no color int Red=1; int Blue=2;  bool bfs(int starting\_vertex){  queue<int> Q;  Q.push(starting\_vertex);  color[starting\_vertex]=Red;  while(!Q.empty()){  int curr=Q.front();  Q.pop();  for(int i=0;i<adj[curr].size();i++){  if(color[adj[curr][i]]==-1){  int next\_vertex = adj[curr][i];  Q.push(next\_vertex);  if(color[curr]==Red)  color[next\_vertex]=Blue;  else{  color[next\_vertex]=Red;  }  }  if(color[curr]==color[next\_vertex])  return false;  }  }  return true; }  bool checkBiPartite(){  for(int i=0;i<V;i++){  if(Colored[i]!=-1) continue;  if(bfs(i)==true) continue;  else return false;  }  return true; } |





**[15th July 2020, 8:30-11:00pm(IST)](Nitesh))**

**[Class: DP]**

* <https://www.geeksforgeeks.org/program-for-nth-fibonacci-number/>
* <https://www.geeksforgeeks.org/binomial-coefficient-dp-9/>
* <https://www.geeksforgeeks.org/find-minimum-number-of-coins-that-make-a-change/>
* <https://www.geeksforgeeks.org/longest-increasing-subsequence-dp-3/>
* Given a tree with N (1<=N<=1e5) nodes. You are given N-1 queries. In each query you are given an edge. When you remove this edge, the component containing this edge divides into two. You have to tell the number of nodes in the larger component after you remove this edge (DSU)

|  |
| --- |
| #include<bits/stdc++.h> using namespace std;  const long long N = 1010;  int dp[N]; // dp[i] is storing my ith fibonacci number   // O(N) long long fib(long long n) {  if(n<=1)  return n;  if(dp[n] != -1)  return dp[n];  dp[n] = fib(n-1) + fib(n-2);  return dp[n]; }  int main() {  long long n, i;  memset(dp,-1,sizeof(dp));  cin>>n;   // Tabulation / Iterative Approach  dp[0] = 0, dp[1] = 1;  for(i=2;i<=n;i++)  dp[i] = dp[i-1] + dp[i-2];  cout<<dp[n]<<endl;    // cout<<fib(n)<<endl;   return 0; }   /\* F[0] = 0, F[1] = 1, F[n] = F[n-1] + F[n-2];    F[2] = F[0] + F[1] = 0 + 1 = 1  F[3] = F[1] + F[2] = 1 + 1 = 2  F[4] = F[2] + F[3] = 1 + 2 = 3    4  3 2  2 1 1 0 1 0   \*/  #include<bits/stdc++.h> using namespace std;  const int N = 1010;  int dp[N][N]; // dp[i][j] will store C(i,j)   // O(N\*N) // Process of storing our answers in recurstion is known as memoization. int nCr(int n, int r) {  if(n<=1 || r==0 || n==r)  return 1;  if(dp[n][r] != -1)  return dp[n][r];  dp[n][r] = nCr(n-1,r-1) + nCr(n-1,r);  return dp[n][r]; }   int nCr\_iterative(int n, int r) {  int i, j;  for(i=0;i<=n;i++) dp[i][0] = 1;  for(i=1;i<=n;i++)  {  dp[i][i] = 1;  for(j=0;j<i;j++)  {  dp[i][j] = dp[i-1][j-1] + dp[i-1][j];  }  }  return dp[n][r]; }   int main() {  int n, r;  memset(dp, -1, sizeof(dp));  cin>>n>>r;  cout<<nCr(n,r)<<endl;   return 0; }   /\*   not defined for r > n and for r, n < 0  defined 0 <= r <= n, n!=0   C(n, 0) = C(n, n) = 1  C(1, 0) = C(1, 1) = 1  C(n, r) = C(n-1, r-1) + C(n-1, r)    C(2, 0) = 1  C(2, 1) = C(1, 0) + C(1, 1) = 1 + 1 = 2  C(2, 2) = 1  \*/  #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  int dp[N], in[N]; // dp[i] => Length of the longest increasing subsequence which ends at index i  // To calculate length of the LIS ending at ith index, look at lengths of all the LIS ending at index j (j<i, in[j]<in[i]) // take maximum among them, 1 + max(LIS(j));     int main() {  int i, n, ans = 1;  cin>>n;  for(i=0;i<n;i++)  cin>>in[i];  for(i=0;i<n;i++)  {  dp[i] = 1;  for(j=i-1;j>=0;j--)  if(in[j] < in[i])  dp[i] = max(dp[i], 1 + dp[j]);  }   for(i=0;i<n;i++)  ans = max(ans, dp[i]);  cout<<ans<<endl;  return 0; }     /\*  LIS: Longest Increasing Subsequence  Given an array, tell the length LIS  I/N: { 2, 10, 3, 4, 1}  dp[0] = 1 , 2 dp[1] = 2 , 2 10 dp[2] = 2 , 2 3 dp[3] = 3 , 2 3 4 dp[4] = 1 , 1    3, 4   3, 4, 8  \*/     #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  int coins[N], n, dp[N]; // dp[i] will store the minimum number of coins required to make sum i. // n => Total number of coins // O(N\*V) // memset(dp, -1, sizeof(dp)); int minCoins(int V) {  if(V==0)  return 0;  if(dp[i] != -1)  return dp[i];  int i, n, ans = 1e9;  for(i=0;i<n;i++)  if(V >= coins[i])  ans = min(ans, minCoins(V-coins[i]));  dp[i] = ans+1;  return dp[i]; }   int minCoins\_iterative(int V) {  int i, j, ans;  dp[0] = 0;  for(i=1;i<=V;i++)  {  ans = 1e9;  for(j=0;j<n;j++)  if(i >= coins[j])  ans = min(ans, dp[i-coins[j]]);  dp[i] = ans+1;  }  return dp[V]; }  /\*  Input: coins[] = {25, 10, 5}, V = 30  You want to make sum 30 if i take 25 now, Find out how many coins you require to make sum 5 => 1 -> 2 if i take 10 now, Find out how many coins you require to make sum 20 => 2 -> 3 if i take 5 now, Find out how many coins you require to make sum 25 => 1 -> 2  You want to make sum 20 if i take 10 now, Find out how many coins you require to make sum 10 => 1 -> 2 if i take 5 now, Find out how many coins you require to make sum 15 => 2 -> 3      func(coins[], V) => minimum coins required to get sum V { return 1 + min(func(coins[], V-25), func(coins[], V-10), func(coins[], V-5)) // 5, 20, 25 // 2, 3, 1 }  \*/ |

**[17th July 2020, 8:30-11:00pm(IST)](Nitesh))**

**[Class: DP, Greedy Problems]**

* <https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/>
* <https://www.geeksforgeeks.org/longest-common-subsequence-dp-4/>
* <https://www.geeksforgeeks.org/edit-distance-dp-5/>

**Greedy Problems:**

* <https://www.geeksforgeeks.org/connect-n-ropes-minimum-cost/>
* [https://www.geeksforgeeks.org/greedy-algorithm-to-find-minimum-number-of-coins/](https://www.geeksforgeeks.org/connect-n-ropes-minimum-cost/)

|  |
| --- |
| #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  string s1, s2; int dp[N][N];  /\*   This function will give me the lcs of s1 and s2 if both of them are considered till indices m and n respectively.   Arguments are last indices of the string;    aaa, aaa => 3  aaa, aa => 2  ab , ba => 1 (a, b)   aba, baa => 2   abb, baa => 1   (abb, ba), (ab, baa)  O(m\*n)  \*/ int lcs(int m, int n) {  if(m<0 || n<0)  return 0;  if(dp[m][n] != -1)  return dp[m][n];  if(s1[m] == s2[n])  dp[m][n] = 1 + lcs(m-1,n-1);  // if last character is same, we can include it in our lcs  else  dp[m][n] = max(lcs(m-1,n), lcs(m,n-1));  // if last character is not same, whatever is the lcs, it won't have last characters of both the strings  return dp[m][n]; }  int main() {  cin>>s1>>s2;  memset(dp, -1, sizeof(dp));  cout<<lcs(s1.size()-1, s2.size()-1)<<endl;  return 0; }   /\*  LCS => Longest Common Subsequence  Given two sequences, find the length of longest subsequence present in both of them. A subsequence is a sequence that appears in the same relative order, but not necessarily contiguous. For example, "abc", "abg", "bdf", "aeg", "acefg", .. etc are subsequences of "abcdefg".  Example LCS for input Sequences "ABCDGH" and "AEDFHR" is "ADH" of length 3. LCS for input Sequences "AGGTAB" and "GXTXAYB" is "GTAB" of length 4.  \*/  #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  int dp[N][N]; string s1, s2;   /\*  minimum number of operations required to convert string s1 to string s2 if both strings are considered till m and n  respectively.  s1: abc s2: abb  2, 2  1 + (1,1) \*/  int edit(int m, int n) {  if(m < 0)  return n+1; // Inserting  if(n < 0)  return m+1; // Deleting  if(dp[m][n] != -1)  return dp[m][n];   if(s1[m] == s2[n])  dp[m][n] = edit(m-1, n-1);  else  dp[m][n] = 1 + min(edit(m, n-1), min(edit(m-1, n), edit(m-1, n-1)));   return dp[m][n]; }  int main() {  cin>>s1>>s2;  memset(dp, -1, sizeof(dp));  cout<<edit(s1.lenght()-1, s2.length-1)<<endl;  return 0; }   /\*   Given two string, s1 and s2, tell the minimum number of operations in which you can convert s1 to s2   In one operation, you can   Insert a character anywhere in s1  Delete any character of s1  replace any character of s1 with any other character.    s1 => m  s2 => n  if(m>n)  m => replace first n characters, delete last m-n characters  else  n => replace fist m character, insert n-m characters in the end       max(m, n)  \*/  #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  int val[N], wt[N], dp[N][N];  /\*   1-based index  considering items from 1st index till nth index and limit of weight is W, how much maximum value we can get.  index = 1, 2, 3   val[] = 100, 60, 120, 1000  wt[] = 20, 10, 30, 60  W = 50   4, 50  3, 50  max(120 + (2, 20), (2, 50)) => 220  (2, 50) => 160  2, 20  max(60 + (1, 10), (1, 20)) => 100  (1, 10) => 0  (1, 20) => 100 \*/  int knapsack(int n, int W) {  if(n==0 || W==0)  return 0;  if(dp[n][W] != -1)  return dp[n][W];  if(wt[n] > W)  dp[n][W] = knapsack(n-1, W);  else  dp[n][W] = max(val[n]+knapsack(n-1, W-wt[n]), knapsack(n-1, W));  return dp[n][W]; }  int main() {  int W, n, i;  cin>>n>>W;  memset(dp, -1, sizeof(dp));  for(i=1;i<=n;i++)  cin>>val[i]>>wt[i];  cout<<knapsack(n, W)<<endl;  return 0; }  #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  int val[N], wt[N], dp[N][N]; // dp[i][j]: Maximum value we can get if we consider till ith item and if the limit of weight is j.   int knapsack(int n, int W) {  int i,j;  for(i=1;i<=n;i++)  {  for(j=1;j<=W;j++)  {  dp[i][j] = dp[i-1][j];  if(wt[i] <= j)  dp[i][j] = max(dp[i][j], val[i] + dp[i-1][j-wt[i]]);  }  }  return dp[n][W]; }   int main() {  int W, n, i;  memset(dp, 0, sizeof(dp));  cin>>n>>W;  for(i=1; i<=n; i++)  cin>>val[i]>>wt[i];  cout<<knapsack(n,W)<<endl;  return 0; } |

**[20th July 2020, 8:30-11:00pm(IST)](Nitesh))**

**[Class: Backtracking, Trie]**

* <https://www.geeksforgeeks.org/rat-in-a-maze-backtracking-2/>
* <https://www.geeksforgeeks.org/subset-sum-backtracking-4/>
* <https://www.geeksforgeeks.org/m-coloring-problem-backtracking-5/>
* <https://www.quora.com/q/threadsiiithyderabad/Tutorial-on-Trie-and-example-problems>

|  |
| --- |
| #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  int in[N][N], n;  // returns true if there is a path from (x, y) to (n-1, n-1), otherwise it returs false bool rec(int x, int y) {  if(x>=n || y>=n || in[x][y] == 0)  return 0;  if(x == n-1 && y == n-1)  return 1;  if(rec(x+1, y) || rec(x, y+1))  return 1;  return 0; }  int main() {  cout<<rec(0, 0)<<endl;  return 0; }  #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  int n, k, in[N], bin[N]; // 0 to n-1  /\* in = {10, 5}   bin: 0, 0   bin: 0, 1   bin: 1, 0   bin: 1, 1  \*/  void rec(int i) {  if(i == n)  {  int j, sum=0;  for(j=0;j<n;j++)  if(bin[j] == 1)  sum+=in[j];  if(sum == k)  {  for(j=0;j<n;j++)  if(bin[j] == 1)  cout<<in[j]<<" ";  cout<<endl;  }  return;  }  bin[i] = 0;  rec(i+1);  bin[i] = 1;  rec(i+1);  return; }  int main() {  cin>>n>>k;  for(int i=0;i<n;i++)  cin>>in[i];  rec(0);  return 0; }  /\* We will make all possible binary sequences of length n. Positions at which 1 is there in the binary sequence, we will include  elements from the array from those positions and will check if there sum is k.  in : {1, 3, 5 ,4} k: 7   0000 0001 0010 0011 0100 0101 0110 0111  \*/ #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  int in[N][N], n, m, color[N];  // return true if ith node can be given color number j at the current situation. int safe(int i, int j) {  for(int k=0;k<n;k++)  if(in[i][k] == 1 && color[k] == j)  return 0;  return 1; }  bool rec(int i) {  if(i == n)  {  for(j=0;j<n;j++)  cout<<color[j]<<endl;  return 1;  }  for(int j = 1;j<=m;j++)  {  if(safe(i, j))  {  color[i] = j;  if(rec(i+1))  return 1;  }  }  return 0; }  int main() {  memset(color, -1, sizeof(color));  cout<<rec(0)<<endl;   return 0; }  #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  int to[N][26], flag[N], cnt = 0; string s;  // to[i][j] -> i is the node number, j represents edge (0-25, a->0, b->1 ..., z->25) // root is represented by i = 0 // if to[i][j] == 0 -> from node number i, edge j does not exists // otherwise to[i][j] stores the node number which we reach if we start from node number i and take edge j.   void insert(string s) {  int cur = 0;  for(int i=0;i<s.size();i++)  {  if(to[cur][s[i]-'a'] == 0)  to[cur][s[i]-'a'] = ++cnt;  cur = to[cur][s[i]-'a'];  }  flag[cur] = 1;  return;  }  bool qry(string s) {  int cur = 0;  for(int i = 0;i<s.size();i++)  if(to[cur][s[i]-'a'] == 0)  return 0;  else  cur = to[cur][s[i]-'a'];  if(flag[cur] == 1)  return 1;  return 0; }  int main() {  int n, q;  cin>>n>>q;  for(i=0;i<n;i++)  {  cin>>s;  insert(s);  }  while(q--)  {  cin>>s;  cout<<qry(s)<<endl;  }  return 0; }   /\*  Given a set of words.  Now answer some queries. In each query a word will be given, tell if this word is present in your set of words.    \*/  #include<bits/stdc++.h> using namespace std;  const long long N = 2010;  int to[N][2], cnt = 0; string s;  // to[i][j] -> i is the node number, j represents edge  // root is represented by i = 0 // if to[i][j] == 0 -> from node number i, edge j does not exists // otherwise to[i][j] stores the node number which we reach if we start from node number i and take edge j.  void insert(string s) {  int cur = 0;  for(int i=0;i<s.size();i++)  {  if(to[cur][s[i]-'0'] == 0)  to[cur][s[i]-'0'] = ++cnt;  cur = to[cur][s[i]-'0'];  }  return;  }   string qry(string s) {  int cur = 0;  string res = "";  for(int i=0; i<s.size();i++)  {  if(s[i] == '0')  {  if(to[cur][1] != 0)  cur = to[cur][1], res+="1";  else  cur = to[cur][0], res+="0";  }  else  {  if(to[cur][0] != 0)  cur = to[cur][0], res+="0";  else  cur = to[cur][1], res+="1";  }  }  return res; }  int main() {  int n, q;  cin>>n>>q;  for(i=0;i<n;i++)  {  cin>>s;  insert(s);   }  while(q--)  {  cin>>s;  cout<<qry(s)<<endl;  }  return 0; }   /\*  1 0 -> 1  0 1 -> 1  0 0 -> 0  1 1 -> 0  Given an array of binary sequences, and given a number k (binary number)  . Return the sequence from array whose xor with k is maximum.  (same length sequences) \*/ |