Algorithm Lab Practice

Created	@June 17, 2023 7:23 PM
Last Edited Time	@June 20, 2023 8:42 AM
By	Borhan

- Binary Search

```
#include<bits/stdc++.h>
using namespace std;

int main(){
  int n;  cin >> n;
  int arr[n];
  for(int i=0; i<n; i++){
    cin >> arr[i];
  }


  int target; cin >> target;

  int i=0, j=n-1;
  int mid = (i+j)/2;
  while(i <= j){
    mid = (i+j)/2;
    if(arr[mid] == target) break;
    if(arr[mid] > target){
      j = mid-1;
    } else{
      i = mid+1;
    }
  }

  if(arr[mid] == target){
    cout << "Found at " << mid << "\n";
  } else {
    cout << "Not found.";
  }
}
```

- Finding the rightmost index if there are multiple same input

```
#include<bits/stdc++.h>
using namespace std;

int main(){
  int n;  cin >> n;
  int arr[n];
  for(int i=0; i<n; i++){
    cin >> arr[i];
  }


  int target; cin >> target;

  int i=0, j=n-1;
  int mid = (i+j)/2;
  int ans = -1;
  while(i <= j){
    mid = (i+j)/2;
    if(arr[mid] == target) {
      ans = mid;
      i = mid+1;
    }
    else if(arr[mid] > target){
      j = mid-1;
    } else{
      i = mid+1;
    }
  }

  if(ans != -1){
    cout << "Found at " << ans << "\n";
  } else {
    cout << "Not found.";
  }
}
```

- Finding the first element that is greater than target

```
#include<bits/stdc++.h>
using namespace std;

int main(){
  int n;  cin >> n;
  int arr[n];
  for(int i=0; i<n; i++){
    cin >> arr[i];
  }
  int target; cin >> target;

  int i=0, j=n-1;
  int mid = (i+j)/2;
  int ans = -1;
  while(i <= j){
    mid = (i+j)/2;
    if(arr[mid] > target) {
      ans = mid;
      j = mid-1;
    }
    else if(arr[mid] > target){
      j = mid-1;
    } else{
      i = mid+1;
    }
  }

  if(ans != -1){
    cout << "the first element that is greater than target " << arr[ans] ;
  } else {
    cout << "Not found.";
  }
}
```

- Binary Search using Recursion

```
#include<bits/stdc++.h>
using namespace std;

#define MAX (int)1e5+5

int n;
int arr[MAX];

int BS(int i, int j, int target){
  if(i>j) return -1;

  int mid = (i+j)/2;

  if(arr[mid]==target) return mid;
  if(target > arr[mid]) return BS(mid+1, j, target);

  return BS(i, mid-1, target);

}
```

```
int main(){
  int n;  cin >> n;
  for(int i=0; i<n; i++){
    cin >> arr[i];
  }
  int target; cin >> target;

  int found = BS(0, n-1, target);

  if(found == -1){
    cout << "Not found";
  } else {
    cout << "Found at " << found;
  }

}
```

- Finding the rightmost index if there are multiple same input

```
#include<bits/stdc++.h>
using namespace std;

#define MAX (int)1e5+5

int n;
int arr[MAX];

int BS(int i, int j, int target){
  if(i>j) return -1;

  int mid = (i+j)/2;

  if(arr[mid]==target) return max(mid, BS(mid+1, j, target));
  if(target > arr[mid]) return BS(mid+1, j, target);

  return BS(i, mid-1, target);

}

int main(){
  int n;  cin >> n;
  for(int i=0; i<n; i++){
    cin >> arr[i];
  }
  int target; cin >> target;

  int found = BS(0, n-1, target);

  if(found == -1){
    cout << "Not found";
  } else {
    cout << "Found at " << found;
  }

}
```

- Heap

```
#include<bits/stdc++.h>
using namespace std;
#define MAX (int)1e5+5

class Heap{
  int arr[MAX+5];
  int size;

  public:
  Heap(){
    arr[0]=-1;
    size=0;
  }

  void print(){
    for(int i=1; i<=size; i++)  cout << arr[i] << " ";
  }

  void push(int n){
    size++;
    if(size >= MAX){
      size--;
      cout << "Heap is full.";
      return;
    }

    int i = size;
    arr[size]=n;

    while(i > 1){
      int parent = i/2;
      if(arr[parent] < arr[i]){
        swap(arr[parent], arr[i]);
        i=parent;
      } else return;
    }
  }
  void pop(){
    arr[1]=arr[size];
    size--;

    int i=1;
    while(i < size){
      bool change=false;
      int left = i*2;
      int right = i*2 + 1;
      if(left <= size && arr[i] < arr[left]){
        swap(arr[i], arr[left]);
        i = left;
        change = true;
      }
      if(right <= size && arr[i] < arr[right]){
        swap(arr[i], arr[right]);
        i=right;
        change = true;
      }
      if(!change) return;
    }
  }
};

int main(){
  int n; cin >> n;
  Heap hp;
  while(n--){
    int x, y; cin >> x;
    if(x==1){
      //push
      cin >> y;
      hp.push(y);
    } else if(x==2){
      // pop
      hp.pop();
    } else{
      //print
      hp.print();
    }
  }
  return 0;
}
```

- Heap Sort

```
#include<bits/stdc++.h>
using namespace std;
```

```
void heapify(int n, int arr[], int size){
  int left = n*2;
  int right = left+1;
  int largest = n;

  if(left <= size && arr[left] > arr[largest]){
    largest = left;
  }
  if(right <= size && arr[right] > arr[largest]){
    largest  = right;
  }

  if(n != largest){
    swap(arr[n], arr[largest]);
    heapify(largest, arr, size);
  }
}

void heapsort(int n, int arr[]){
  int size = n;

  for(int i=n/2; i>0; i--){
    heapify(i, arr, n);
  }

  while(size > 1){
    swap(arr[1], arr[size]);
    size--;
    heapify(1, arr, size);
  }
}

int main(){
  int n;  cin >> n;
  int arr[n+1];
  arr[0]=-1;

  for(int i=1; i<=n; i++) cin >> arr[i];

  heapsort(n, arr);

  for(int i=1; i<=n; i++) cout << arr[i] << " ";

  return 0;
}
```

- Merging Two Sorted Array

```
#include<bits/stdc++.h>
using namespace std;

int main(){
  int n;  cin >> n;
  int a[n];
  for(int i=0; i<n; i++)  cin >> a[i];

  int m;  cin >> m;
  int b[n];
  for(int i=0; i<m; i++) cin >> b[i];

  vector<int> c;
  int i=0, j=0;
  while(i<n || j < m){
    if(i < n && j < m){
      if(a[i] < b[j]){
        c.push_back(a[i]);
        i++;
      } else {
        c.push_back(b[j]);
        j++;
      }
    } else if(i < n){
      c.push_back(a[i]);
      i++;
    } else{
      c.push_back(a[j]);
      j++;
    }
  }

  for(int i=0; i<n+m; i++) cout << c[i] << " ";

}
```

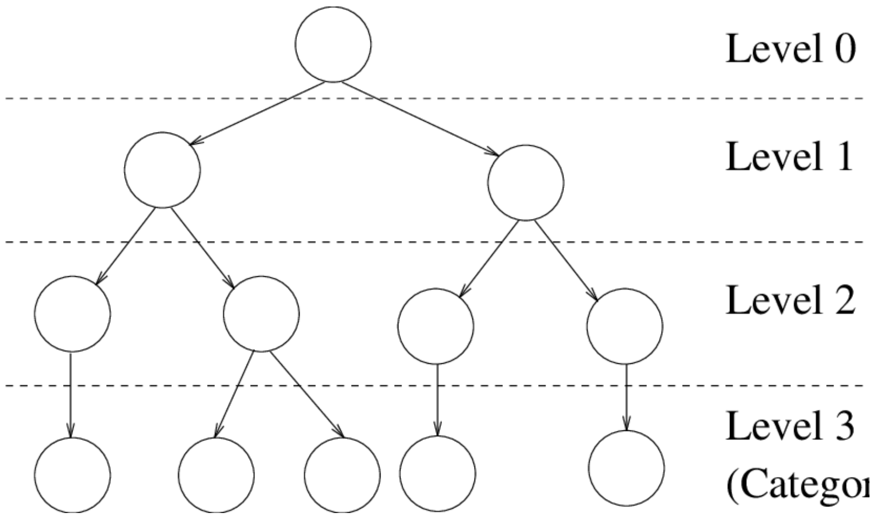
- Breadth First Searching

```
#include<bits/stdc++.h>
using namespace std;

int main(){
  int n, e; cin >> n >> e;
  vector<int> adj[n+5];

  for(int i=1; i<=e; i++){
    int u, v; cin >> u >> v;
    adj[u].push_back(v);
    adj[v].push_back(u);
  }

  int source; cin >> source;

  queue<int> q;
  vector<int> ans;
  vector<bool> vis(n+2, false);

  vis[source] = true;
  q.push(source);
  ans.push_back(source);

  while(!q.empty()){
    int u = q.front();
    q.pop();

    for(auto v:adj[u]){
      if(vis[v]) continue;
      vis[v]=true;
      ans.push_back(v);
      q.push(v);
    }
  }

  for(auto l:ans) cout << l << " ";
}
```

- Shortest Path Between Two Node by BFS

```
#include <bits/stdc++.h>
using namespace std;

int main(){
  int n,e;  cin >> n >> e;

  vector<int> adj[n+2];
  for(int i=1; i<=n; i++){
    int u, v; cin >> u >> v;
    adj[u].push_back(v);
    adj[v].push_back(u);
  }

  int src, dst; cin >> src >> dst;

  vector<int> parent(n+1, -1);
  vector<bool> vis(n+2, false);
  queue<int> q;

  q.push(src);
  vis[src] = true;

  while(!q.empty()){
    int u = q.front();
    q.pop();

    for(auto v:adj[u]){
      if(vis[v]) continue;
      parent[v]=u;
      vis[v] = true;
      q.push(v);
    }
  }
  vector<int> ans;
  for(int i=dst; i != -1; i=parent[i]){
    ans.push_back(i);
  }

  reverse(ans.begin(), ans.end());
  for(auto l:ans) cout << l << " ";
}
```

- Maximum Level of a Graph by BFS



```
#include <bits/stdc++.h>
using namespace std;

const int MAX = 1000005;
```

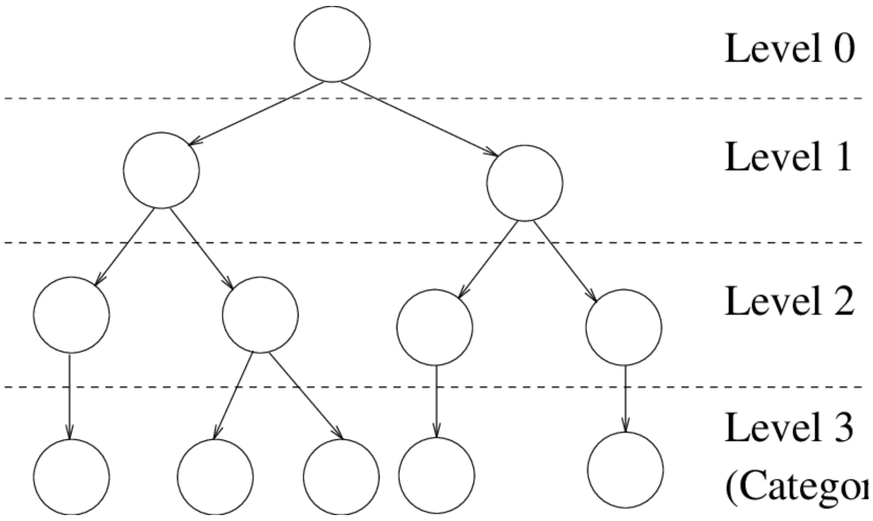
```
vector<int> adj[MAX+5];
int ans = INT_MIN;

void bfs(int i, vector<int> &vis){
  queue<pair<int, int>> q;
  q.push({i,0});
  vis[i]=true;

  while(!q.empty()){
    auto [u, level] = q.front();
    q.pop();
    ans = max(ans, level);
    for(auto v:adj[u]){
      if(!vis[v]){
        vis[v] = true;
        q.push({v, level+1});
      }
    }
  }
}

int main(){
  int n,e;  cin >> n >> e;
  while(e--){
    int u, v; cin >> u >> v;
    adj[u].push_back(v);
    adj[v].push_back(u);
  }

  vector<int> vis(n+2, false);

  for(int i=1; i<=n; i++){
    if(!vis[i]){
      bfs(i, vis);
    }
  }
  cout << ans;
}
```

- Depth First Searching

```
#include<bits/stdc++.h>
using namespace std;

const int MAX = (int)1e5+5;

vector<int> adj[MAX+5];
```

```
vector<int> ans;

void dfs(int u, vector<bool> &vis){
  vis[u] = true;
  ans.push_back(u);
  for(auto v:adj[u]){
    if(vis[v]) continue;
    dfs(v, vis);
  }
}

int main(){
  int n, e; cin >> n >> e;
  for(int i=1; i<=e; i++){
    int u, v; cin >> u >> v;

    adj[u].push_back(v);
    adj[v].push_back(u);
  }

  vector<bool> vis(n+5, false);
  dfs(1, vis);

  for(auto l:ans) cout << l << " ";
}
```

- Maximum Level of a Graph Using DFS



```
#include <bits/stdc++.h>
using namespace std;
const int MAX = 1000005;

vector<int> adj[MAX+5];

int ans = INT_MIN;

void dfs(int u, vector<int> &vis, int level){
  vis[u] = true;
  ans = max(ans, level);
  level++;
  for(auto v:adj[u]){
    if(!vis[v]) dfs(v, vis, level);
  }
}


int main(){
  int n,e;  cin >> n >> e;
  while(e--){
    int u, v; cin >> u >> v;
    adj[u].push_back(v);
    adj[v].push_back(u);
  }

  vector<int> vis(n+2, false);
  for(int i=1; i<=n; i++){
    if(!vis[i]) dfs(i, vis, 0);
  }

  cout << ans;
}
```

- Priority Queue

```
priority_queue<int> pq;
pq.push(); // To insert
pq.pop();  // Delete the topmost element
pq.top();  // Return the topmost elemenet
```

- Max Heap by Priority Queue

```
priority_queue<int>
```

- Min Heap by Priority Queue

```
// If you want to store integer data
priority_queue<int, vector<int>, greater<int>> pq;

// If you want to store pair
priority_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>>> pq;
```

- Sorting in Ascending order using Priority Queue

```
#include <bits/stdc++.h>
using namespace std;

int main(){
  priority_queue<int, vector<int>, greater<int>> pq;
  int n;  cin >> n;

  for(int i=1; i<=n; i++){
    int x;  cin >> x;
    pq.push(x);
  }

  while(!pq.empty()){
    cout << pq.top() << " ";
    pq.pop();
  }

  return 0;
}
```

- Dijkstra

```
#include <bits/stdc++.h>
using namespace std;

int main(){
  int n,e;  cin >> n >> e;

  vector<pair<int, int>> adj[n+2];
  for(int i=1; i<=e; i++){
    int u, v, w;  cin >> u >> v >> w;

    adj[u].push_back({v, w});
    adj[v].push_back({u, w});
  }

  int src, dst;   cin >> src >> dst;

  vector<bool> vis(n+2, false);

  vector<int> dis(n+2, INT_MAX);
  dis[src]=0;

  priority_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>>> pq;
  pq.push({0, src});

  while(!pq.empty()){
    pair<int, int> u = pq.top();
    pq.pop();

    int distance = u.first;
    int node = u.second;
    vis[node] = true;

    for(pair<int, int> v: adj[node]){
      int node2 = v.first;
      int distance2 = v.second;
      if(vis[node2]) continue;
      if(dis[node2] > distance+distance2){
        dis[node2] = distance+distance2;
        pq.push({dis[node2], node2});
      }
    }
  }

  cout << dis[dst] << " ";
}
```

- Dijkstra by Matrix

```
#include <bits/stdc++.h>
using namespace std;

int main(){
  int n,e;  cin >> n >> e;
  int adj[n+2][n+2];

  for(int i=0; i<=n; i++){
    for(int j=0; j<=n; j++)
      adj[i][j] = -1;
  }

  while(e--){
    int u, v, w;  cin >> u >> v >> w;
    adj[u][v] = w;
    adj[v][u] = w;
  }
  int src, dst; cin >> src >> dst;

  vector<int> dist(n+2, INT_MAX);
  vector<bool> vis(n+2, false);
  vis[src] = 1;
  dist[src] = 0;

  priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
  pq.push({0, src});

  while(!pq.empty()){
    auto [d_u, u] = pq.top();
    pq.pop();

    vis[u] = true;
    for(int i=1; i<=n; i++){

      if(i==u) continue;
      if(adj[u][i] == -1) continue;
      if(vis[i]) continue;

      if(dist[i] > adj[u][i] + d_u){
        dist[i] = adj[u][i] + d_u;
        pq.push({dist[i], i});
      }
    }
  }

  cout << dist[dst];
}
```

- Fibonacci by Recursive DP

```
#include <bits/stdc++.h>
using namespace std;

#define MAX 1000005
```

```
long long  dp[MAX+5];

long long fib(long long n){
  if(n < 0) return 0;
  if(n==0) return dp[n]=0;
  if(n==1) return dp[n]=1;
  if(dp[n] != -1) return dp[n];
  long long ans = fib(n-1) + fib(n-2);
  return dp[n] = ans;
}

int main(){
  long long  n; cin >> n;
  memset(dp, -1, sizeof(dp));

  //Printing the n'th Fibonacci Number
  cout << fib(n);

  //Printing the 1-n'th Fibonacci Number
  cout << "\n";
  for(int i=1; i<=n; i++){
    cout << dp[i] << " ";
  }

  return 0;
}
```

- Fibonacci by Iterative DP

```
#include <bits/stdc++.h>
using namespace std;

int main(){
  long long  n; cin >> n;

  long long dp[n+10];
  dp[0] = 0; dp[1] = 1;

  for(int i=2; i<=n; i++){
    dp[i] = dp[i-1] + dp[i-2];
  }

  //Printing the n'th Fibonacci Number
  cout << dp[n];

  //Printing the 1-n'th Fibonacci Number
  cout << "\n";
  for(int i=1; i<=n; i++){
    cout << dp[i] << " ";
  }

  return 0;
}
```

- Knapsack

```
#include <bits/stdc++.h>
using namespace std;

const int MAX = (int)1e4;

int n;
vector<int> cost(MAX+5);
vector<int> weight(MAX+5);
int dp[MAX+5][MAX+5];
int func(int n, int w){
  if(n < 1 || w < 1) return 0;
  if(dp[n][w] != -1) return dp[n][w];

  int ans = func(n-1, w);

  if (w - weight[n]  >= 0)
  ans = max(cost[n] + func(n-1, w - weight[n]), ans);

  return dp[n][w] = ans;
}


int main(){
  memset(dp, -1, sizeof(dp));

  int n, w; cin >> n >> w;
  for(int i=1; i<=n; i++) cin >> weight[i];
  for(int i=1; i<=n; i++) cin >> cost[i];

  cout << func(n, w);
}
```