|  |  |
| --- | --- |
| **LAB Assignment - 4** | |
| NAME: **Aashirwad Kumar** | Roll NUMBER: **IMH/10004/18** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Question – 1 – a) | Implement the 0/1 Knapsack problem using Dynamic Programming method | | |  |
| CODE  #include <bits/stdc++.h>  using namespace std;  #define fast ios::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  typedef long long ll;typedef long double ld;typedef pair<int,int> pii;  #define F first  #define S second  #define PB push\_back  #define MP make\_pair  const ll mod = 1e9+7, N = 2e6+7, M = 2e6+7, INF = INT\_MAX/10;  ll powe(ll x, ll y){ x = x%mod, y=y%(mod-1);ll ans = 1;while(y>0){if (y&1){ans = (1ll \* x \* ans)%mod;}y>>=1;x = (1ll \* x \* x)%mod;}return ans;}  int findMax(vector<int>&items , vector<int>&weights , int capacity )  {  int n=items.size();  vector<vector<int>>dp(n+1,vector<int>(capacity+1,0));  for(int i=1 ; i<=n ; i++)  {  int ite = items[i-1];  int wt = weights[i-1];  for(int j=1 ; j<=capacity ; j++)  {  if(wt<=j)  {  dp[i][j] = max(dp[i-1][j-wt]+ite , dp[i-1][j]);  }  else  {  dp[i][j] = dp[i-1][j];  }  }  }  return dp[n][capacity];  }  int main()  {  vector<int> items{60,100,120};  vector<int> weights{10,20,30};  int capacity=50;  int maxValues =findMax(items , weights , capacity);  cout<<"max values knapsack dp : " <<maxValues;  cout<<"\n";  #ifndef ONLINE\_JUDGE  cout<<"\nTime Elapsed : " << 1.0\*clock() / CLOCKS\_PER\_SEC << " s\n";  #endif  return 0;  } | | | | |
| OUTPUT  1output.JPG | | | | |
| Question -1 – b) | Implement the 0/1 Knapsack problem using Greedy method. | | |  |
| CODE  #include <bits/stdc++.h>  using namespace std;  #define fast ios::sync\_with\_stdio(0);cin.tie(0);cout.tie(0);  typedef long long ll;typedef long double ld;typedef pair<int,int> pii;  #define F first  #define S second  #define PB push\_back  #define MP make\_pair  const ll mod = 1e9+7, N = 2e6+7, M = 2e6+7, INF = INT\_MAX/10;  ll powe(ll x, ll y){ x = x%mod, y=y%(mod-1);ll ans = 1;while(y>0){if (y&1){ans = (1ll \* x \* ans)%mod;}y>>=1;x = (1ll \* x \* x)%mod;}return ans;}  void solve\_prg(){    int n;  cin>>n;    int x,y;  vector<pair<int,int>> v;  for(int i=0;i<n;i++){  cin>>x>>y;  v.push\_back({x,y});  }  sort(v.begin(),v.end());    int weight;  cin>>weight;  vector <pair<int,int>> bag;  int wt\_bag=weight;  int val\_bag=0;  for(int i=0;i<n;i++){  if(v[i].second<=wt\_bag){  bag.push\_back(v[i]);  wt\_bag-=v[i].second;  val\_bag+=v[i].first;  }  else{  for(int j=0;j<bag.size();j++){  if(wt\_bag+bag[j].second>=v[i].second && bag[j].first<v[i].first){  wt\_bag+=bag[j].second;  wt\_bag-=v[i].second;  val\_bag-=bag[j].first;  val\_bag+=v[i].first;  bag[j].second=v[i].second;  bag[j].first=v[i].first;  break;  }  }  }  }  cout<<"Problem Optimum value "<<val\_bag<<endl;  }  int main() {    solve\_prg();  #ifndef ONLINE\_JUDGE  cout<<"\nTime Elapsed : " << 1.0\*clock() / CLOCKS\_PER\_SEC << " s\n";  #endif  return 0;  } | | | | |
| OUTPUT  1boutput.JPG | | | | |
| Question - 2 | | From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm. |  | |
| CODE  #include<bits/stdc++.h>  using namespace std;  vector<pair<int, int>> adj[7];  vector<bool> v(7,false);  vector<int> dis(7, INT\_MAX);  priority\_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>>>pq;  void dji\_algo(){    pq.push({0,1});  dis[1]=0;    while(!pq.empty()){    auto front = pq.top().second;  v[front]=true;  pq.pop();    for(auto i:adj[front]){    int wgt=i.second;  int node = i.first;    if(!v[node] && dis[front] + wgt < dis[node]) {  dis[node]=dis[front]+wgt;  pq.push({dis[node], node});  }  }      }    }  void print() {  for (int i = 1; i <=6; i++){  cout<<"From Node 1"<<" to "<<i<<" is "<<dis[i]<<endl;  }    }  int main()  {  adj[1].push\_back({2, 3});  adj[2].push\_back({1, 3});  adj[1].push\_back({4, 7});  adj[4].push\_back({1, 7});  adj[2].push\_back({5, 11});  adj[5].push\_back({2, 11});  adj[3].push\_back({1, 3});  adj[1].push\_back({3, 3});  adj[3].push\_back({2,7});  adj[2].push\_back({3,7});  adj[5].push\_back({1,3});  adj[1].push\_back({5,3});  adj[5].push\_back({4,5});  adj[4].push\_back({5,5});  adj[5].push\_back({6,7});  adj[6].push\_back({5,7});  adj[6].push\_back({4,1});  adj[4].push\_back({6,1});    cout<<"Single Source Shortest Paths "<<endl;    dji\_algo();    print();  cout<<endl;  #ifndef ONLINE\_JUDGE  cout<<"\nTime Elapsed : " << 1.0\*clock() / CLOCKS\_PER\_SEC << " s\n";  #endif    return 0;  } | | | | |
| OUTPUT  2ouput.JPG | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Question- 3 –a) | | Implement All-Pairs Shortest Paths problem using Floyd's algorithm. |  | |
| CODE  #include<bits/stdc++.h>  using namespace std;  vector<pair<int, int>> adj[7];  vector<vector<int>> adj\_mat(6,vector<int>(6,INT\_MAX));  void fw\_algo(vector<vector<int>> &adj\_mat){    for (int k = 0; k < 6; k++)  {  for (int i = 0; i < 6; i++)  {  for (int j = 0; j < 6; j++)  {  if (adj\_mat[i][k] + adj\_mat[k][j] < adj\_mat[i][j] && (adj\_mat[k][j] != INT\_MAX && adj\_mat[i][k] != INT\_MAX)){  adj\_mat[i][j] = adj\_mat[i][k] + adj\_mat[k][j];  }    }  }  }      }  void print() {    for (int i = 0; i < 6; i++) {  for (int j = 0; j < 6; j++) {  if (adj\_mat[i][j] == INT\_MAX){  cout<<"INF"<<" ";  }  else{  cout<<adj\_mat[i][j]<<" ";  }    }  cout<<endl;  }    }  int main()  {  adj[1].push\_back({2, 3});  adj[2].push\_back({1, 3});  adj[1].push\_back({4, 7});  adj[4].push\_back({1, 7});  adj[2].push\_back({5, 11});  adj[5].push\_back({2, 11});  adj[3].push\_back({1, 3});  adj[1].push\_back({3, 3});  adj[3].push\_back({2,7});  adj[2].push\_back({3,7});  adj[5].push\_back({1,3});  adj[1].push\_back({5,3});  adj[5].push\_back({4,5});  adj[4].push\_back({5,5});  adj[5].push\_back({6,7});  adj[6].push\_back({5,7});  adj[6].push\_back({4,1});  adj[4].push\_back({6,1});  for(int i=1;i<=6;i++){  adj\_mat[i-1][i-1]=0;    for(auto j:adj[i]){    int sec = j.first;  int val = j.second;  adj\_mat[i-1][sec-1] = val;  }  }  cout<<"All Sources Shortest Paths using Floyd Warshall "<<endl;    fw\_algo(adj\_mat);    print();    return 0;  } | | | | |
| OUTPUT  3output.JPG | | | | |
| Question – 3-b) | Implement Travelling Sales Person problem using Dynamic programming. | | |  |
| CODE    #include<bits/stdc++.h>  using namespace std;  vector<pair<int, int>> adj[5];  vector<vector<int>> adj\_mat(4,vector<int>(4,INT\_MAX));  int dp[20][10];  int n = 4;  int compute(int ma\_in,int curr,int &ma){  if(ma\_in==ma){  return adj\_mat[curr][0];  }    if(dp[ma\_in][curr]!=-1){  return dp[ma\_in][curr];  }    int ans = INT\_MAX;    for(int i=0;i<n;i++){  if((ma\_in&(1<<i))==0) {  int temp = adj\_mat[curr][i] + compute(ma\_in|(1<<i), i,ma);  ans = min(ans, temp);    }  }    dp[ma\_in][curr] = ans;  return dp[ma\_in][curr];      }  int main()  {  adj[1].push\_back({2, 3});  adj[2].push\_back({1, 3});  adj[1].push\_back({4, 7});  adj[4].push\_back({1, 7});  adj[3].push\_back({1, 3});  adj[1].push\_back({3, 3});  adj[3].push\_back({2,7});  adj[2].push\_back({3,7});  adj[2].push\_back({4,5});  adj[4].push\_back({2,5});  adj[3].push\_back({4,6});  adj[4].push\_back({3,6});  for(int i=1;i<=4;i++){  adj\_mat[i-1][i-1]=0;    for(auto j:adj[i]){    int sec = j.first;  int val = j.second;  adj\_mat[i-1][sec-1] = val;  }  }  cout<<"Minimum Weight Sum for a Travelling Salesman is"<<endl;    for(int i=0;i<(1<<n);i++){  for(int j=0;j<n;j++){  dp[i][j] = -1;  }  }  int ma = (1<<n) - 1;  cout<<compute(1,0,ma);  return 0;  } | | | | |
| OUTPUT  4output.JPG | | | | |