**LIS:**

#include <bits/stdc++.h>

using namespace std;

#define ll long long int

int find\_lis(vector<int> a)

{

vector<int> dp;

for (int i : a)

{

int pos = lower\_bound(dp.begin(), dp.end(), i) - dp.begin();

if (pos == dp.size())

{

dp.push\_back(i);

}

else

{

dp[pos] = i;

}

}

return dp.size();

}

int main()

{

int n;

cin >> n;

vector<int> num(n);

for (int i = 0; i < n; i++)

{

cin >> num[i];

}

int ans = find\_lis(num);

cout << ans << endl;

return 0;

}

**0/1 Knapsack:**

#include <iostream>

#include <vector>

using namespace std;

#define ll long long int

int main()

{

int n, ks;

cin >> n >> ks;

vector<int> item(n), price(n);

for (int i = 0; i < n; i++)

{

cin >> item[i];

}

for (int i = 0; i < n; i++)

{

cin >> price[i];

}

vector<vector<int>> ans(n + 1, vector<int>(ks + 1, 0));

for (int i = 1; i <= n; i++)

{

for (int j = 0; j <= ks; j++)

{

if (j < item[i - 1])

{

ans[i][j] = ans[i - 1][j];

}

else

{

ans[i][j] = max(ans[i - 1][j], ans[i - 1][j - item[i - 1]] + price[i - 1]);

}

}

}

ll cnt = 0;

for (int i = 0; i < n; i++)

{

cnt += price[i];

}

if (cnt == ans[n][ks])

{

cout << "My King, I am successful in capturing the big fish. Immortality is few steps away." << endl;

}

else

{

cout << "My King, I have captured " << ans[n][ks] << " followers till now and I need more soldiers asap." << endl;

}

return 0;

}

**BFS Traversal:**

#include <bits/stdc++.h>

using namespace std;

void bfs(vector<vector<int>> &adjMatrix, vector<bool> &visited, vector<int> &ans, int vertex, int start)

{

queue<int> q;

q.push(start);

visited[start] = true;

while (!q.empty())

{

int frontNode = q.front();

q.pop();

ans.push\_back(frontNode);

for (int i = 1; i <= vertex; i++)

{

if (adjMatrix[frontNode][i] == 1 && !visited[i])

{

q.push(i);

visited[i] = true;

}

}

}

}

vector<int> BFS(vector<vector<int>> &adjMatrix, int vertex, int start)

{

vector<int> ans;

vector<bool> visited(vertex+1, false);

bfs(adjMatrix, visited, ans, vertex, start);

return ans;

}

int main()

{

int vertex;

cin >> vertex;

vector<vector<int>> adjMatrix(vertex+1, vector<int>(vertex+1, 0));

// Input the adjacency matrix

for (int i = 1; i <= vertex; i++)

{

for (int j = 1; j <= vertex; j++)

{

cin >> adjMatrix[i][j];

}

}

int start;

cout << "Enter the starting point for BFS: ";

cin >> start;

vector<int> ans = BFS(adjMatrix, vertex, start);

cout << "BFS traversal from " << start << ": ";

for (int i = 0; i < ans.size(); i++)

{

cout << ans[i] << " ";

}

cout << endl;

return 0;

}