作业3-2018302141224-吴健豪：

李老师您好！这是我的第二次作业

您如果觉得看起来不方便可以查看我的GitHub：

<https://github.com/rushingfox/AlgorithmHomework/tree/master/homework3>

**1.**

|  |
| --- |
| #include <iostream> |
|  | #include <algorithm> |
|  | #include <vector> |
|  | using namespace std; |
|  |  |
|  | vector<int> A = {3,4,3,1}; |
|  | vector<int> B = {1,2,2,6}; |
|  | //use greedy algorithm: though the min requirement meet, it still needs a room(unless the maximum of all the rooms still can not accomodate this meeting ,but if so other meetings can not be accommodated too), so allocate the min size room can ensure that the rest rooms are the biggest in each location in an ordered suquence |
|  | //使用贪心算法，无论如何需求人数最小的会议都会要一间房间（除非没有一个满足，而这种情况给其他会议也无法有房间满足），而分配给他最小的房间能够保证在所有情况中（指分配给最小会议非最小房间的情况）剩余房间情况最好（按照大小排序，每个位置上都是大于等于其他情况的这个位置上的房间的大小） |
|  | int main() |
|  | { |
|  | int m = A.size(); |
|  | int n = B.size(); |
|  | sort(A.begin(), A.end()); |
|  | sort(B.begin(), B.end()); |
|  | int result = 0; |
|  | int p = 0; |
|  | for (int i = 0;i<m&&p<n;i++) |
|  | { |
|  | for (int j = p;j<n;j++) |
|  | { |
|  | if (A[i]<=B[j]) |
|  | { |
|  | result += 1; |
|  | p = j + 1; |
|  | break; |
|  | } |
|  | } |
|  | } |
|  | cout << result << endl; |
|  | system("pause"); |
|  | } |

**2.**

|  |
| --- |
| #include <iostream> |
|  | using namespace std; |
|  | const int m = 2; |
|  | const int n = 2; |
|  |  |
|  | int main() |
|  | { |
|  | int dp[m][n]; |
|  | for (int i = 0; i < m; i++) |
|  | { |
|  | for (int j = 0;j<n;j++) |
|  | { |
|  | if (i==0||j==0) |
|  | { |
|  | dp[i][j] = 1; |
|  | } |
|  | else |
|  | { |
|  | dp[i][j] = dp[i - 1][j] + dp[i][j - 1]; |
|  | } |
|  | } |
|  | } |
|  | cout << dp[m-1][n-1]; |
|  | system("pause"); |
|  | return dp[m - 1][n - 1]; |
|  | } |

**3.**

|  |
| --- |
| #include<iostream> |
|  | #include<vector> |
|  | using namespace std; |
|  |  |
|  | const string arr1 = "apple"; |
|  | const string arr2 = "peach"; |
|  | const int m = 5; |
|  | const int n = 5; |
|  | vector<char> result; |
|  |  |
|  | void solvedp(int(&dp)[m + 1][n + 1]); |
|  |  |
|  | int main() |
|  | { |
|  | int dp[m+1][n+1]; |
|  | solvedp(dp); |
|  | int i = m ; |
|  | int j = n ; |
|  | while (i!=0||j!=0) |
|  | { |
|  | if (i>=1&&dp[i-1][j]==dp[i][j]) |
|  | { |
|  | result.push\_back(arr1[i - 1]); |
|  | i-=1; |
|  | } |
|  | else if (j>=1&&dp[i][j-1]==dp[i][j]) |
|  | { |
|  | result.push\_back(arr2[j - 1]); |
|  | j -= 1; |
|  | } |
|  | else if (i>=1&&j>=1&&dp[i][j-1]!=dp[i][j]&&dp[i-1][j]!=dp[i][j])//´ËÊ±µÄarr1[i-1]ºÍarr2[j-1]ÏàµÈ |
|  | { |
|  | result.push\_back(arr1[i - 1]); |
|  | i -= 1; |
|  | j -= 1; |
|  | } |
|  | } |
|  | for (int i = result.size()-1; i>=0; i--) |
|  | { |
|  | cout << result[i]; |
|  | } |
|  | system("pause"); |
|  | return 0; |
|  | } |
|  |  |
|  | void solvedp(int (&dp)[m+1][n+1]) |
|  | { |
|  | for (int i = 0; i < m + 1; i++) |
|  | { |
|  | for (int j = 0; j < n + 1; j++) |
|  | { |
|  | if (i==0||j==0) |
|  | { |
|  | dp[i][j] = 0; |
|  | } |
|  | else |
|  | { |
|  | if (arr1[i - 1] == arr2[j - 1]) |
|  | { |
|  | dp[i][j] = dp[i - 1][j - 1] + 1; |
|  | } |
|  | else |
|  | { |
|  | dp[i][j] = dp[i][j - 1] > dp[i - 1][j] ? dp[i][j - 1] : dp[i - 1][j]; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |

**4.**

图算法设计章节中涉及的四个算法中，Dijkstra算法不适合负权，因而排除。

其他的三个算法都适合负权无环图。将原图中每个边权值取相反数，再使用SPFA、贝尔曼福特、弗洛伊德算法均能算出最短路径，即是原图的最长路径。