Cubes and Cylinders

by xennygrimmato

Submitted a few seconds ago • Score: 30.00 Status: Accepted

✔ Test Case #0	✔ Test Case #1	✔ Test Case #2
✔ Test Case #3	✔ Test Case #4	✔ Test Case #5
✔ Test Case #6	✔ Test Case #7	✔ Test Case #8
✔ Test Case #9	✔ Test Case #10	✔ Test Case #11
✔ Test Case #12	✔ Test Case #13	✔ Test Case #14
✔ Test Case #15	✔ Test Case #16	✔ Test Case #17
✔ Test Case #18	✔ Test Case #19	✔ Test Case #20
✔ Test Case #21	✔ Test Case #22	✔ Test Case #23
✔ Test Case #24		

Submissions by Friends

Rank	User	Score	School
1	NiceBuddy	25.242	Universität Stuttgart
...			

Full Contest Leaderboard

Submitted Code

Language: C++14 Open in editor

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4  const float root = sqrt(2);
5
6  inline void swap(int *xp, int *yp)
7  {
8      int temp = *xp;
9      *xp = *yp;
10     *yp = temp;
11 }
12
13 inline void bubbleSort(vector<int> &S, vector<int> &K, unsigned int n)
14 {
15     int i, j;
16     for (i = 0; i < n-1; i++)
17         for (j = 0; j < n-i-1; j++)
18             if (S[j] > S[j+1])
19                 {
```

```
                swap(&S[j], &S[j+1]);
                swap(&K[j], &K[j+1]);
            }
}
vector<string> split_string(string);/** n              n             m               m */
inline int maximumPackages(vector<int> &S, vector<int> &K, vector<int> &R, vector<int> &C)
{
    int count = 0;
    float LHS = 0.0;
    float RHS = 0.0;

    bubbleSort(S,K, S.size());
    bubbleSort(R,C, R.size());

    for(unsigned int i = 0; i<S.size(); ++i)
    {
        int curEdge = S[i];
        int currNo  = K[i];

        for(unsigned int j = 0; j<R.size(); ++j)
        {
            int currRadius  = R[j];
            int currCapa    = C[j];

            if( (C[j] !=0 ) && (currNo != 0 ) )
            {
                LHS = static_cast<float>(currRadius)*2;
                RHS = static_cast<float>(curEdge)*root;

                if(LHS>RHS)
                {
                    if(currNo > currCapa)
                    {
                        count+=currCapa;
                        C[j] = 0;
                        currNo = currNo - currCapa;
                    }
                    else    // currNo < currCapa
                    {
                        count+=currNo;
                        C[j] = C[j] - currNo;
                        currNo = 0;
                    }
                }
            }
        }
    }

    return count;
}


int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);     cout.tie(nullptr);

    string nm_temp;
    getline(cin, nm_temp);

    vector<string> nm = split_string(nm_temp);
    int n = stoi(nm[0]);
    int m = stoi(nm[1]);

    /** ******************************************/
    string S_str_temp;      getline(cin, S_str_temp);
    vector<string> S_str = split_string(S_str_temp);
    vector<int> S(n);
    for (int S_i = 0; S_i < n; S_i++)
    {
        int S_item = stoi(S_str[S_i]);
        S[S_i] = S_item;
    }
    /** ******************************************/

    string K_str_temp;      getline(cin, K_str_temp);
    vector<string> K_str = split_string(K_str_temp);
    vector<int> K(n);
```

```
    for (int K_i = 0; K_i < n; K_i++)
    {
        int K_item = stoi(K_str[K_i]);
        K[K_i] = K_item;
    }
    /** ******************************************/

    //cin.ignore(numeric_limits<streamsize>::max(), '\n');
    string R_str_temp;      getline(cin, R_str_temp);
    vector<string> R_str = split_string(R_str_temp);
    vector<int> R(m);
    for (int R_i = 0; R_i < m; R_i++)
    {
        int R_item = stoi(R_str[R_i]);
        R[R_i] = R_item;
    }
    /** ******************************************/

    string C_str_temp;      getline(cin, C_str_temp);
    vector<string> C_str = split_string(C_str_temp);
    vector<int> C(m);
    for (int C_i = 0; C_i < m; C_i++)
    {
        int C_item = stoi(C_str[C_i]);
        C[C_i] = C_item;
    }

    /** ******************************************/
    int result = maximumPackages(S, K, R, C);

    cout << result << "\n";

    return 0;
}

inline vector<string> split_string(string input_string)
{
    string::iterator new_end = unique(input_string.begin(), input_string.end(), [] (const char &x, const char &y) {
        return x == y and x == ' ';
    });

    input_string.erase(new_end, input_string.end());

    while (input_string[input_string.length() - 1] == ' ') {
        input_string.pop_back();
    }

    vector<string> splits;
    char delimiter = ' ';

    size_t i = 0;
    size_t pos = input_string.find(delimiter);

    while (pos != string::npos) {
        splits.push_back(input_string.substr(i, pos - i));

        i = pos + 1;
        pos = input_string.find(delimiter, i);
    }

    splits.push_back(input_string.substr(i, min(pos, input_string.length()) - i + 1));

    return splits;
}

```