Making A Large Island (/problems/making-a-large-island/)

Submission Detail

75 / 75 test cases passed.

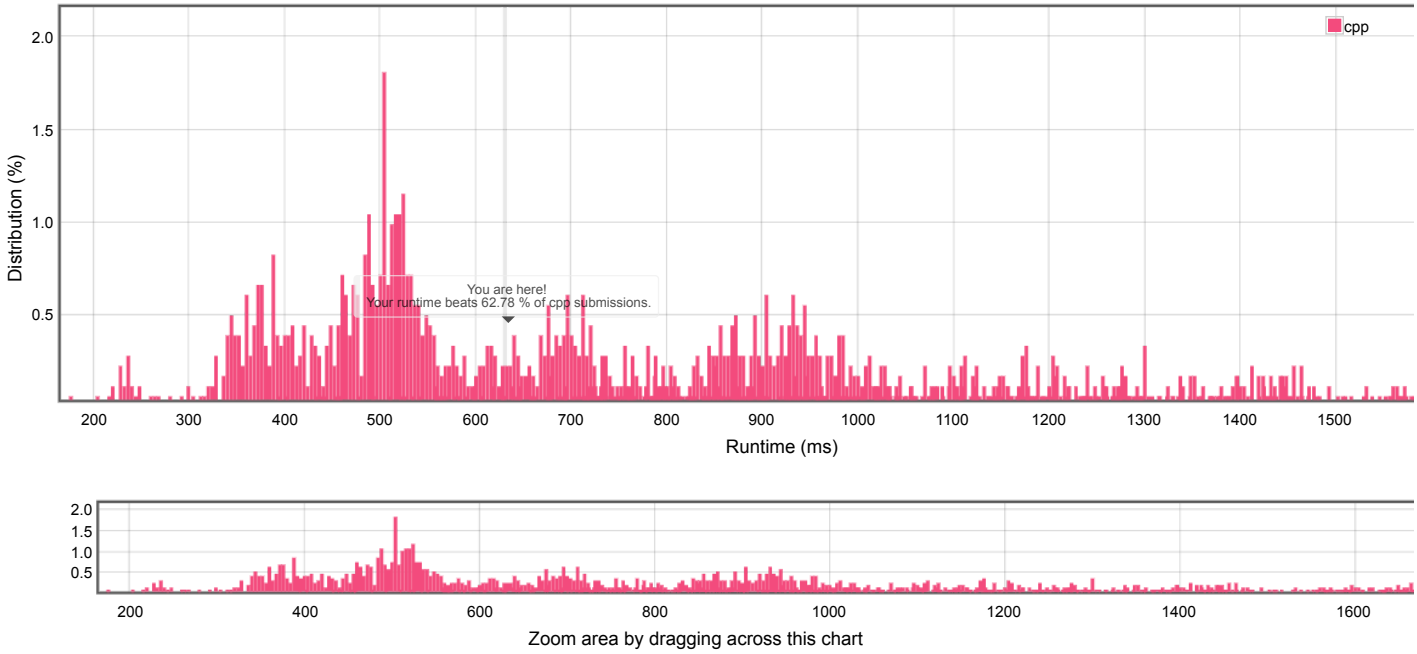
Status: Accepted

Runtime: **632 ms**

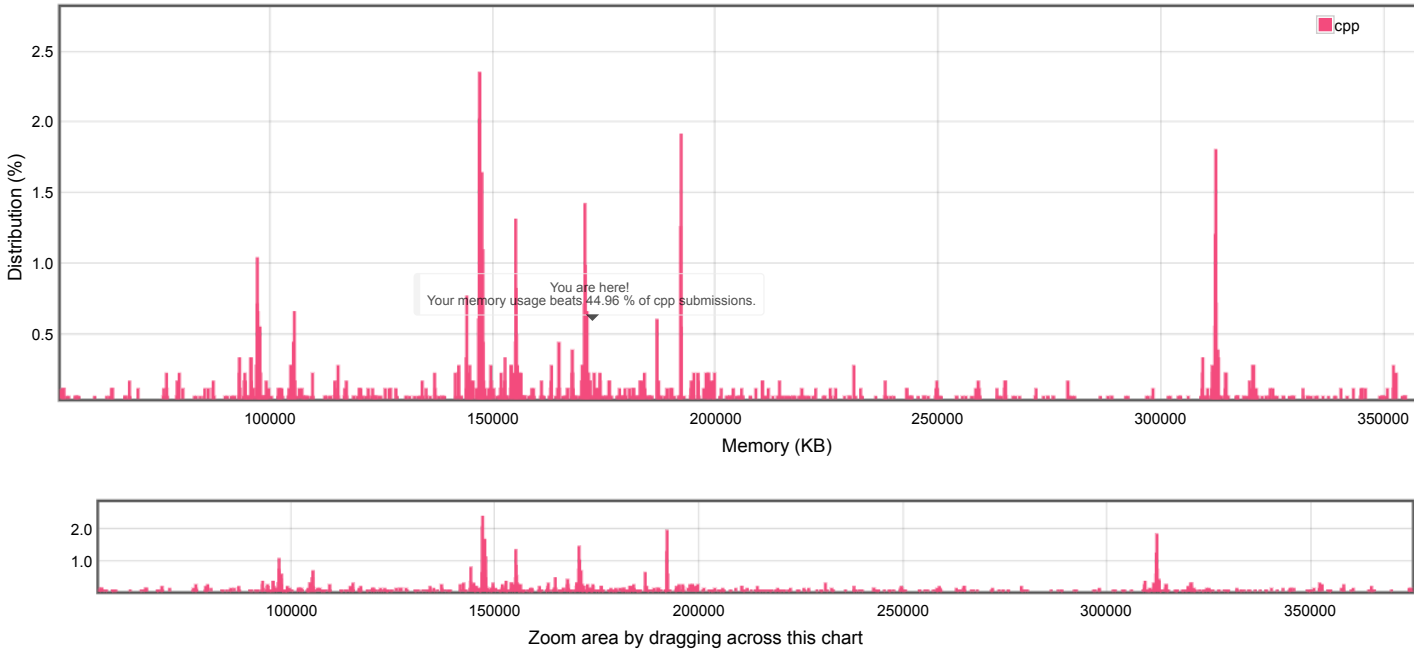
Memory Usage: **171.8 MB**

Submitted: **0 minutes ago**

Accepted Solutions Runtime Distribution



Accepted Solutions Memory Distribution



Invite friends to challenge **Making A Large Island**

Submitted Code: 0 minutes ago

Language: cpp

Edit Code

```
1  class Solution {
2  public:
```

```
    int n,m;
    /*almacena el tamaño de cada componente conectado asignando un ID a cada celda del componente conectado que actuará como una clave en el m
    std::unordered_map<int, int> tam;
    /*almacena -1 para la celda que no se visito y la identificación de celda para la visitada*/
    int A[502][502];


    // islandNum es el ID
    int dfs(int i, int j, std::vector<std::vector<int>>& grid, int islandNum){
        if(i<0 || i>=n || j<0 || j>=m || grid[i][j]==0 || A[i][j]!=-1){
            return 0;
        }
        /*todos los componentes conectados tienen la misma identificación*/
        A[i][j]=islandNum;
        int tam=1;
        tam+=dfs(i+1,j,grid,islandNum);
        tam+=dfs(i-1,j,grid,islandNum);
        tam+=dfs(i,j+1,grid,islandNum);
        tam+=dfs(i,j-1,grid,islandNum);
        return tam;
    }

    int largestIsland(std::vector<std::vector<int>>& grid) {
        n=grid.size();
        m=grid[0].size();
        memset(A,-1,sizeof(A));
        int l=1;
        for(int i=0;i<n;i++){
            for(int j=0;j<m;j++){
                if(grid[i][j]==1 && A[i][j]==-1){
                    tam[l]=dfs(i, j, grid, l);
                    l++;
                }
            }
        }

        int maxarea=0;
        for(int i=0;i<n;i++){
            for(int j=0;j<m;j++)
            {
              /*si encontramos un 0, intentaremos encontrar los componentes conectados con diferentes IDs*/
                if(grid[i][j]==0)
                {
                    int area=1;
                    std::set<int> s;
                    if(i+1<n)
                    s.insert(A[i+1][j]);
                    if(i-1>=0)
                    s.insert(A[i-1][j]);
                    if(j+1<m)
                    s.insert(A[i][j+1]);
                    if(j-1>=0)
                    s.insert(A[i][j-1]);
                    for(int c: s){
                        area+=tam[c]; // agregando tam para cada componente conectado único
                    }
                    maxarea=std::max(maxarea, area);
                }
            }
        }
        if(maxarea!=0)
        return maxarea;
        return n*m;
    }
};
```

Back to problem (/problems/making-a-large-island/)