MANGO THIEF

****
Author: Aditya Raj
****

Explaination:
For solving this question ,

The basic approch would be like, for each child you search all the tree having boundary less than the jumping capacity of child and then total the mangoes .In this way you will get the no. Of mangoes a child can have.

You have to do this for every child and this will give O(n^2) time complexity.

But here we want to do it in less time . So we will use sorting and binary serach in this way we can do it in O(n logn).

The approch would be
1. sort all the mango tree according to their height (use pair for taking height and mangoes at same time).
2. take the prefix (sum from start upto that point) sum in an array .So that when you know location that upto which child can steal the . mango, you don't have to dot he sum again and again.(as it will again become O(n^2).
3. just do binary search and print the prefix sum.

Code: (in c++)

```
#include<bits/stdc++.h>
using namespace std;

int main(){
	int n,m,i,j,a,b,c;
	cin>>n>>m;
	int ch[n];
	pair<int,int> mango[m];
	for(i=0;i<n;i++)
		cin>>ch[i];
	for(i=0;i<m;i++){
		cin>>a>>b;
		mango[i]=make_pair(a,b);
	}
	//sorting the mango with height
	sort(mango,mango+m);
	int cum[m]={0};

	// taking the prefix sum
	for(i= 0;i<m;i++){
		if(i){
			cum[i] = cum[i-1] + mango[i].second;
		}
		else
```

```
                cum[i] = mango[i].second;
        }
        //doing the binary serach
        for(i=0;i<n;i++){

                int low = 0 , high = m-1;
                int ans = 0;
                while(low <= high){

                        int mid = low + ( (high- low)/2);

                        if(mango[mid].first <= ch[i]){
                                ans = cum[mid];
                                low = mid + 1;
                        }
                        else
                                high = mid - 1;
                }
                cout<<ans<<" ";
        }
        cout<<endl;
}
```

Time Complexity : O(n logn)