

Set

Set은 중복을 허용하지 않는 자료구조이다.

- Set은 unordered_set과 ordered_set으로 나뉜다.
- unordered_set은 원소의 순서를 보장하지 않는다.
- ordered_set은 원소의 순서를 보장한다.
- Set은 원소를 추가, 삭제, 검색을 할 수 있다.

Set의 특징

1. 원소

2. 중복 허용 여부

3. 순서

4. 원소 추가, 삭제, 검색 방법

5. Set의 종류

- unordered_set: 원소의 순서를 보장하지 않는다.
- ordered_set: 원소의 순서를 보장한다.
- set: 원소의 순서를 보장한다.

6. Set의 사용법

7. Set의 예제

- Set을 선언하고 원소를 추가하는 방법
- Set에서 원소를 검색하는 방법

8. Set의 성능

- unordered_set: 원소 추가, 삭제, 검색의 성능이 O(1)이다.
- ordered_set: 원소 추가, 삭제, 검색의 성능이 O(log n)이다.
- set: 원소 추가, 삭제, 검색의 성능이 O(log n)이다.

9. Set의 응용

10. Set의 문제풀이

11. Set의 정리

- unordered_set: `set<int> s;`
- ordered_set: `s.insert(3);`
- set: `set<int> s;`
 - unordered_set과 pair를 사용한다.
 - `set<pair<int, int>>`
- Set의 원소를 검색하는 방법
 - `for(auto x: s) x.first < s.first`
 - `pair<int, int>`
 - `x.first`
 - `x.second`
- `s.begin()->first`

- `s.begin()` 返回迭代器
- `s.end()` `s.begin()` 的下一个位置
- `s.erase()` 删除元素
- `s.size()` 返回大小

快速排序


 alt text

- 快速排序 -> 递归
- 快速排序 -> 分治
- 快速排序的时间复杂度是 $O(n^2)$
- 快速排序的空间复杂度是 $O(1)$

```
class KthLargest {
public:
    typedef pair<int,int> PII;
    int tot,k;
    set<PII> s;
    KthLargest(int k, vector<int>& nums) {
        this->k = k; //this指向当前对象
        for(auto x: nums)
        {
            add(x);
        }
        return;
    }

    int add(int val) {
        if(s.size() < k)
        {
            s.insert(PII(val,tot++));
        }
        else{
            if(s.begin()->first < val)
            {
                s.insert(PII(val,tot++));
            }
        }
        if(s.size() > k) s.erase(s.begin());
        return s.begin()->first;
    }
};
```

快速排序

 alt text

- 快速排序
- 快速排序

- 00000000
 - 00000000000000000000
 - 0000000000000000000000000000000000
 - 00000000000000000000
 - 0000000000000000
 - 00000000
 - 00000000
- 00000000000000000000000000000000001
- n1000000000000 0010000000000000

```

class MedianFinder {
public:
    typedef pair<int,int> PII;
    int tot;
    set<PII> s1,s2;
    MedianFinder() {
        tot = 0;
    }

    void addNum(int num) {
        if(s1.size() == 0 || num < -s1.begin()->first)
            s1.insert(PII(-num,tot++));
        else
            s2.insert(PII(num,tot++));
        int n1 = (s1.size() + s2.size() + 1) / 2;
        if(n1 == s1.size()) return;
        if(s1.size() < n1)
        {
            s1.insert(PII(-s2.begin()->first,tot++));
            s2.erase(s2.begin());
        }
        else
        {
            s2.insert(PII(-s1.begin()->first,tot++));
            s1.erase(s1.begin());
        }
    }


    double findMedian() {
        if((s1.size() + s2.size()) % 2)
            return -s1.begin()->first;
        double a = -s1.begin()->first;
        double b = s2.begin()->first;
        return (a + b) / 2.0;
    }
};

```



```
class Solution {
public:
    int nthUglyNumber(int n) {
        set<long long> s;
        s.insert(1);
        long long ans = 0;
        while(n-->0)
        {
            ans = *s.begin();
            s.erase(s.begin());
            if(ans % 5 == 0)
            {
                s.insert(ans * 5);
            }
            else if(ans % 3 == 0)
            {
                s.insert(ans * 3);
                s.insert(ans * 5);
            }
            else
            {
                s.insert(ans * 2);
                s.insert(ans * 3);
                s.insert(ans * 5);
            }
        }
        return ans;
    }
};
```

□□□□

 alt text

- [illegible]

```
#include<bits/stdc++.h>
using namespace std;
struct Data
{
    int p,d;
    Data(int p,int d):p(p),d(d){}
```

```



    bool operator<(const Data &obj) const
    {
        if(d != obj.d) return d < obj.d;
        return p > obj.p;
    }
};

typedef pair<int,int> PII;

int main()
{
    int n;
    cin >> n;
    vector<Data> arr;
    set<PII> s;
    for(int i = 0,p,d;i < n;i++)
    {
        cin >> p >> d;
        arr.push_back(Data(p,d));
    }
    sort(arr.begin(),arr.end());
    for(int i = 0;i < n;i++)
    {
        if(arr[i].d > s.size())
            s.insert(PII(arr[i].p,i));
        else
        {
            if(arr[i].p > s.begin()->first)
            {
                s.erase(s.begin());
                s.insert(PII(arr[i].p,i));
            }
        }
    }
    int ans = 0;
    for(auto x: s)
        ans += x.first;
    cout << ans;
    return 0;
}

```

☐☐M☐☐

 alt text ☐☐☐☐☐☐☐☐i☐☐☐M☐☐☐☐☐i-1☐☐☐i☐m☐☐☐☐☐☐m * m☐☐☐☐  alt text

```

#include<bits/stdc++.h>
using namespace std;
typedef pair<int,int> PII;
int main()
{

```

```

int n,m,t = 0;
cin >> n >> m;
set<PII> s;
s.insert(PII(0, t++));
for(int i = 0;i < n;i++)
{
    vector<int> temp;
    for(auto x : s)
    {
        temp.push_back(x.first);
    }
    s.clear();
    for(int j = 0 ,a;j < m;j++)
    {
        cin >> a;
        for(auto x: temp)
        {
            if(s.size() < m || s.begin()->first < x - a)//□□□□□□
                s.insert(PII(x - a,t++));
            if(s.size() > m) s.erase(s.begin());
        }
    }
}
int flag = 0;
for(auto iter = s.rbegin();iter != s.rend();iter++)//□□□□□
{
    if(flag) cout << " " ;
    cout << -iter->first ;
    flag = 1;
}
return 0;
}

```