ACM-ICPC 导论

魏 通

苏州大学 • 计算机科学与技术学院

2016 年 1 月 30 日

Complexity (算法复杂度)

Big O notation

$$complexity \begin{cases} time & complexity \\ space & complexity \end{cases}$$

pronunciation: Big oh...

$O(1) \quad O(n) \quad O(n \times \log^n) \quad O\left(n^2\right) \cdots$

$1s \approx 10^7$

Precomputation (预处理)

A simple example

Sample 1

给一个 $n\,(1 \le 10^6)$ 个整数的数组，有 $Q\,(Q \le 10^6)$ 次询问，每次询问包含两个整数 L 和 R, 输出该区间的和。

Solution

$$令\ sum[i] = \sum_{j=1}^{i} a_j，\ 则 \sum_{i=L}^{R} a_i = sum[R] - sum[L-1]$$

Another example

Sample 2

给出一个 1000000 范围的正整数，求其不同的素因子的个数。

Conclusion

空间换时间

Greedy (贪心算法)

Sample 3

有 1 元，5 元，10 元硬币各 C1，C5, C10 枚，现在要用这些硬币来支付 A 元，最少需要多少枚硬币？假定本题至少存在一种支付方案。

$0 \leq A \leq 10^9$

$0 \leq C_i \leq 10^9$

Bit manipulation (位操作)

Code 1: BIT 中的 lowbit

```
int lowbit(int x) {
  return x & -x;
}
```

集合的整数表示

1. $1 << i$ 只含有第 i 个元素的集合 $\{i\}$
2. $(1 << n) - 1$ 含有全部 n 个元素的集合 $\{0, 1, ..., n-1\}$
3. $\mathrm{S} >> i \,\&\, 1$ 判断第 i 个元素是否属于集合 S
4. $\mathrm{S} \mid (1 << i)$ 向集合中加入第 i 个元素
5. $\mathrm{S} \,\&\, \sim (1 << i)$ 从集合中去除第 i 个元素
6. $\mathrm{S} \mid \mathrm{T}$ 集合 S 和 T 的交集
7. $\mathrm{S} \,\&\, \mathrm{T}$ 集合 S 和 T 的并集

Code 2: 子集枚举

```
// 将集合 {0, 1, ...., n - 1} 的所有子集枚举出来
for (int S = 0; S < (1<<n); ++S) {
  // 对子集的处理
}
```

C++ STL (C++ 标准模板库)

Container

1. vector
2. queue or priority_queue
3. map
4. set
5. multimap
6. multiset
7. unordered_set (C++ 11)
8. unordered_map (C++ 11)

Algorithm

1. sort
2. lower_bound
3. upper_bound

Code 3: Usage of STL

```
#include <set>
#include <iostream>
#include <algorithm>

int main(void) {
  std::set<int> s;
  s.insert(4);
  s.insert(10);
  s.insert(2);
  std::set<int>::iterator it = s.find(3);
  if (it != s.end()) std::cout << (*it) << std::endl;
  std::set<int>::iterator it2 = s.lower_bound(3);
  if (it2 != s.end()) std::cout << (*it2) << std::endl;
}
```

Code 4: Usage of STL

```
int main(void) {
    std::vector<int> vec;
    vec.emplace_back(5);
    vec.emplace_back(1);
    vec.emplace_back(2);
    std::sort(begin(vec), end(vec));

    int pos = lower_bound(begin(vec), end(vec), 3) -
        begin(vec);
    if (pos != (int)vec.size()) {
        std::cout << vec[pos] << std::endl;
    }

    return 0;
}
```

Two Point (尺取法)

Sample

Sample 4

N positive integers ($10 < N < 100000$), each of them less than or equal 10000, and a positive integer S ($S < 100000000$) are given. Write a program to find the minimal length of the subsequence of consecutive elements of the sequence, the sum of which is greater than or equal to S. if no answer, print 0.

Sample Input

$n = 10$

$S = 15$

a $= 5, 1, 3, 5, 10, 7, 4, 9, 2, 8$

Sample Output

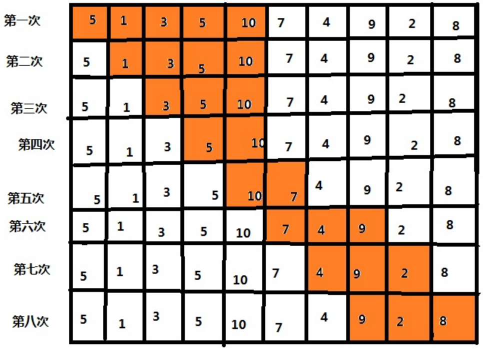
2

Code 5: Silver Solution

```
int n, S;
int a[MAXN];
int sum[MAXN + 1];

void solve() {
  for (int i = 0; i < n; ++i) {
    sum[i + 1] = sum[i] + a[i];
  }

  if (sum[n] < S) {
    printf(''0\n'');
    return;
  }

  int res = n;
```

```
    for (int i = 0; sum[i] + S <= sum[n]; ++s) {
        int t = lower_bound(sum + i, sum + n, sum[i] + S) -
            sum;
        res = min(res, t - s);
    }

    printf("%d\n", res);
}
```

Two Point


第一次
5 1 3 5 10 7 4 9 2 8
第二次
5 1 3 5 10 7 4 9 2 8
第三次
5 1 3 5 10 7 4 9 2 8
第四次
5 1 3 5 10 7 4 9 2 8
第五次
5 1 3 5 10 7 4 9 2 8
第六次
5 1 3 5 10 7 4 9 2 8
第七次
5 1 3 5 10 7 4 9 2 8
第八次
5 1 3 5 10 7 4 9 2 8

Code 6: Golden Solution

```
void solve() {
  int res = n + 1;
  int s = 0, t = 0, sum = 0;
  for ( ; ; ) {
    while (t < n && sum < S) {
      sum += a[t++];
    }
    if (sum < S) break;
    res = min(res, t - s);
    sum -= a[s++];
  }

  if (res > n) res = 0;
  printf("%d\n", res);
}
```

Golden Solution

Because t only moves from 0 to n, so time complexsity is $O(n)$

SG Games (博弈)

Bash Game(巴什博奕)

Sample 5

有一堆石子共 n 个，Alice 和 Bob 轮流取石子，每次至多取 k 个，至少取 1 个，Alice 先取，将石子取完的人获胜。当双方都采取最优策略时，谁会获胜？

Sample Input:

$n = 7, k = 3$

Sample Output:

Alice

Theory 1

必败态：无论取多少个石子都只能转移到必胜态。

必胜态：存在一种取法，可以转移到必败态。

$n=7,k=3$ 当 $n=0$ 时为必败态，因为当前玩家无子可取。而 $n=1$ or $n=2$ or $n=3$ 都可以通过取一定的石子到达必败态，所以他们都是必胜态。

当 $n=4$ 时，无论取 1 or 2 or 3 个石子，都只能到达必胜态，所以他为必败态。

以此类推：$n=(k+1)*i\,(i=1,2,3\cdots)$ 为必败态，其他均为必胜态。

Nim

Sample 6

有 $n(1 \leq 1000000)$ 堆石子，每堆各有 $a_i\left(1 \leq a_i \leq 10^9\right)$ 个石子，Alice 和 Bob 轮流从非空的石子堆中取走至少一颗石子。Alice 先取，将所有石子取完的人获胜。当双方都采取最优策略时，谁会获胜？

Sample Input:

$n = 3$

$a = \{1, 2, 4\}$

Sample Output:

Alice

Theory 2

当 $a_1 \wedge a_2 \wedge \cdots \wedge a_n \neq 0$ 时必胜

当 $a_1 \wedge a_2 \wedge \cdots \wedge a_n = 0$ 时必败

必败态证明

$$a_1 \ \hat{}\ a_2 \ \hat{}\ \cdots \ \hat{}\ a_i \ \hat{}\ \cdots \ \hat{}\ a_n = 0$$

$$a_1 \ \hat{}\ a_2 \ \hat{}\ \cdots \ \hat{}\ a_i' \ \hat{}\ \cdots \ \hat{}\ a_n = 0$$

$$a_i = a_i' \text{ (与题意矛盾)}$$

必胜态证明

$$a_1 \mathbin{\hat{}} a_2 \mathbin{\hat{}} \cdots \mathbin{\hat{}} a_i \mathbin{\hat{}} \cdots \mathbin{\hat{}} a_n = k\,(k \neq 0)$$

$a_i' = a_i \mathbin{\hat{}} k < a_i$(必然存在一个 a_i 和 k 的二进制最高位相同)

$$a_1 \mathbin{\hat{}} a_2 \mathbin{\hat{}} \cdots \mathbin{\hat{}} a_i' \mathbin{\hat{}} \cdots \mathbin{\hat{}} a_n = k \mathbin{\hat{}} a_i \mathbin{\hat{}} a_i' = 0$$

The end
Thank you!