

UoA CP Lecture 3

Sorting and Brute Force

Qingchuan(Sam) Zhang
qzha536@aucklanduni.ac.nz

April 17, 2020

Contents

1	How to sort	2
1.1	Quicksort	2
1.2	<code>std::sort</code>	3
1.2.1	Basic Usage	3
1.2.2	User-defined <code>struct</code> and Customized Comparison Functions	4
2	Sorted, so what?	5
2.1	Binary Search - Sum of Two Values	5
2.2	Stick Lengths	6
3	Just brute force it	7
3.1	<code>for</code> loop – UVa 725	7
3.2	Depth-First Search(DFS)	8
3.2.1	Generating Subset - Apple Division	8
3.2.2	Backtrack - Chessboard and Queens	9
3.2.3	Meet in the middle – Codeforces 888E	10
3.3	Bitmask	11
3.3.1	Generating Subset - Apple Division	11

1 How to sort

1.1 Quicksort

The main idea of this method is:

1. Choose a pivot, randomly.
2. Divide the numbers into three sub-arrays: smaller than pivot, equal to pivot, larger than pivot.
3. Recursively sort the sub-arrays.
4. Merge the sub-arrays.

```
1 vector<int> my_sort(vector<int> nums) {
2     if(nums.size() <= 1) {
3         return nums;
4     } else {
5         // 1. Randomly choose a pivot
6         int pivot_index = rand() % nums.size();
7         auto pivot = nums[pivot_index];
8
9         // 2. Divide the numbers into three sub-arrays
10        vector<int> small, equal, large;
11        for(auto v:nums) {
12            if(v < pivot)small.push_back(v);
13            else if(v == pivot)equal.push_back(v);
14            else if(v > pivot)large.push_back(v);
15        }
16
17        // 3. Recursively sort the sub-arrays
18        small = my_sort(small);
19        //equal = my_sort(equal);
20        large = my_sort(large);
21
22
23        // 4. Merge the sub-arrays
24        vector<int> result;
25        result.insert(result.end(), small.begin(), small.end());
26        result.insert(result.end(), equal.begin(), equal.end());
27        result.insert(result.end(), large.begin(), large.end());
28        return result;
29    }
30 }
```

1.2 std::sort

1.2.1 Basic Usage

```
1  int main() {
2      vector<int> v = {4, 2, 5, 3, 5, 8, 3};
3      // small -> large
4      sort(v.begin(), v.end());
5
6      // large -> small
7      sort(v.rbegin(), v.rend());
8
9      // sort an C-style array
10     int n = 7; // array size
11     int a[] = {4, 2, 5, 3, 5, 8, 3};
12     sort(a, a + n);
13
14
15     // sort a string
16     string s = "monkey";
17     sort(s.begin(), s.end());
18
19     return 0;
20 }
```

1.2.2 User-defined struct and Customized Comparison Functions

```
1  struct P {
2      int x, y;
3
4      bool operator<(const P &p) const {
5          if(x != p.x) {
6              return x < p.x;
7          } else {
8              return y < p.y;
9          }
10     }
11 };
12
13 bool cmp(const P &a, const P &b) {
14     if(a.y != b.y) {
15         return a.y < b.y;
16     } else {
17         return a.x < b.x;
18     }
19 }
20
21 int main() {
22     vector<P> v;
23     v.push_back({1, 2});
24     v.push_back({2, 1});
25     // small -> large
26     sort(v.begin(), v.end());
27     // {(1,2), (2,1)}
28     sort(v.begin(), v.end(), cmp);
29     // {(2,1), (1,2)}
30     return 0;
31 }
```

2 Sorted, so what?

2.1 Binary Search - Sum of Two Values

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main() {
5      int n, target;
6      cin >> n >> target;
7      vector<pair<int, int>> v(n);
8      int cnt = 1;
9      for(auto &i:v) {
10         cin >> i.first;
11         i.second = cnt++;
12     }
13     sort(begin(v), end(v));
14     for(auto itr = begin(v); itr != end(v); itr++) {
15         auto itr2 = lower_bound(begin(v), itr, make_pair(target - itr->first, 0));
16         if(itr2 != itr and itr2->first + itr->first == target) {
17             cout << itr->second << " " << itr2->second << endl;
18             return 0;
19         }
20     }
21     cout << "IMPOSSIBLE" << endl;
22     return 0;
23 }
```

2.2 Stick Lengths

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  typedef long long ll;
5
6  int main() {
7      ll n;
8      cin >> n;
9      vector<ll> v(n);
10     for(auto &i:v)
11         cin >> i;
12
13
14     sort(begin(v), end(v));
15
16     //find the median
17     ll opt = v[n / 2];
18     ll ans = 0;
19     for(int i = 0; i < n; i++)
20         ans += abs(v[i] - opt);
21     cout << ans;
22
23 }
```

3 Just brute force it

3.1 for loop – UVa 725

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  void solve(int n) {
4      bool found = false;
5      for(int a = 0; a < 10; a++) {
6          for(int b = 0; b < 10; b++) {
7              for(int c = 0; c < 10; c++) {
8                  for(int d = 0; d < 10; d++) {
9                      for(int e = 0; e < 10; e++) {
10                         int numerator = a * 10000 + b * 1000 + c * 100 + d * 10 + e;
11                         if(numerator % n == 0) {
12                             int denominator = numerator / n;
13                             string nu = to_string(numerator), de = to_string(denominator);
14                             while(nu.size() < 5) nu = "0" + nu;
15                             while(de.size() < 5) de = "0" + de;
16                             set<char>s;
17                             for(auto i:nu)s.insert(i);
18                             for(auto i:de)s.insert(i);
19                             if(s.size() == 10){
20                                 found = true;
21                                 cout<<nu<<" / "<<de<<" = "<<n<<endl;
22                             }
23                         }
24                     }
25                 }
26             }
27         }
28     }
29     if(!found) cout << "There are no solutions for " << n << "." << endl;
30 }
31 int main() {
32     int n; bool first = true;
33     while(cin >> n) {
34         if(n == 0) {
35             return 0;
36         } else {
37             if(first) first = false;
38             else cout << endl;
39             solve(n);
40         }
41     }
42 }
```

3.2 Depth-First Search(DFS)

3.2.1 Generating Subset - Apple Division

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  typedef long long ll;
5
6  ll ans = LLONG_MAX, num[20], sum = 0;
7  int n;
8
9  void dfs(int step, ll sub_sum) {
10     if(step == n) {
11         ans = min(ans, abs(sub_sum - (sum - sub_sum)));
12     } else {
13         // choose
14         dfs(step + 1, sub_sum + num[step]);
15
16         // not choose
17         dfs(step + 1, sub_sum + 0);
18     }
19 }
20
21 int main() {
22     cin >> n;
23     for(int i = 0; i < n; i++) {
24         cin >> num[i];
25         sum += num[i];
26     }
27     dfs(0,0);
28     cout<<ans<<endl;
29
30     return 0;
31 }
```


3.2.2 Backtrack - Chessboard and Queens

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int n = 8;
5  string maze[n];
6  int pos[n];
7
8  int ans = 0;
9
10 void dfs(int row = 0) {
11     if(row == n)
12         ans++;
13     else {
14         for(pos[row] = 0; pos[row] < 8; pos[row]++) {
15             bool ok = true;
16             if(maze[row][pos[row]] == '*')ok = 0;
17             for(int past = 0; past < row; past++) {
18                 if(pos[past] == pos[row])ok = false;
19                 if(pos[past] - pos[row] == past - row)ok = false;
20                 if(pos[row] - pos[past] == past - row)ok = false;
21             }
22             if(ok) {
23                 dfs(row + 1);
24             }
25         }
26     }
27 }
28
29
30 int main() {
31     for(int i = 0; i < n; i++) {
32         cin >> maze[i];
33     }
34     dfs();
35     cout << ans << endl;
36     return 0;
37 }
```

3.2.3 Meet in the middle – Codeforces 888E

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int n, m;
4  void dfs(vector<int> &num, vector<int> &result, int sum = 0, int cur = 0) {
5      if(cur == num.size()) {
6          result.push_back(sum);
7      } else {
8          // not choose
9          dfs(num, result, (sum + num[cur]) % m, cur + 1);
10
11         // choose
12         dfs(num, result, sum, cur + 1);
13     }
14 }
15 int main() {
16     cin >> n >> m;
17     vector<int> result[2], num[2];
18     for(int i = 0; i < n; i++) {
19         int v;
20         cin >> v;
21         if(i % 2 == 0) {
22             num[0].push_back(v);
23         } else {
24             num[1].push_back(v);
25         }
26     }
27     for(int i = 0; i < 2; i++) {
28         dfs(num[i], result[i]);
29         sort(begin(result[i]), end(result[i]));
30     }
31     int ans = 0;
32     for(auto a: result[0]) {
33         auto iterator = lower_bound(begin(result[1]), end(result[1]), m - a);
34         if(iterator != begin(result[1])) {
35             --iterator;
36             ans = max(ans, a + (*iterator));
37         }
38         ans = max(ans, (a + result[1].back()) % m);
39     }
40     cout << ans << endl;
41     return 0;
42 }
```

3.3 Bitmask

3.3.1 Generating Subset - Apple Division

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  typedef long long ll;
5
6  ll ans = LLONG_MAX, num[20], sum = 0;
7  int n;
8
9
10 int main() {
11     cin >> n;
12     for(int i = 0; i < n; i++) {
13         cin >> num[i];
14         sum += num[i];
15     }
16     for(int mask = 0; mask < (1 << n); mask++) {
17         ll sub_sum = 0;
18         for(int bit = 0; bit < n; bit++) {
19             if((1 << bit) & mask)
20                 sub_sum += num[bit];
21         }
22         ans = min(ans, abs(sub_sum - (sum - sub_sum)));
23     }
24
25
26     cout << ans << endl;
27
28     return 0;
29 }
```