UoA CP Lecture 3
Sorting and Brute Force

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

April 17, 2020

Contents

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.

```
vector<int> my_sort(vector<int> nums) {
    if(nums.size() <= 1) {
        return nums;
    } else {
        // 1. Randomly choose a pivot
        int pivot_index = rand() % nums.size();
        auto pivot = nums[pivot_index];

        // 2. Divide the numbers into three sub-arrays
        vector<int> small, equal, large;
        for(auto v:nums) {
            if(v < pivot)small.push_back(v);
            else if(v == pivot)equal.push_back(v);
            else if(v > pivot)large.push_back(v);
        }

        // 3. Recursively sort the sub-arrays
        small = my_sort(small);
        //equal = my_sort(equal);
        large = my_sort(large);


        // 4. Merge the sub-arrays
        vector<int> result;
        result.insert(result.end(), small.begin(), small.end());
        result.insert(result.end(), equal.begin(), equal.end());
        result.insert(result.end(), large.begin(), large.end());
        return result;
    }
}
```

1.2 std::sort

1.2.1 Basic Usage

```cpp
int main() {
    vector<int> v = {4, 2, 5, 3, 5, 8, 3};
    // small -> large
    sort(v.begin(), v.end());

    // large -> small
    sort(v.rbegin(), v.rend());

    // sort an C-style array
    int n = 7; // array size
    int a[] = {4, 2, 5, 3, 5, 8, 3};
    sort(a, a + n);


    // sort a string
    string s = "monkey";
    sort(s.begin(), s.end());

    return 0;
}
```

1.2.2 User-defined `struct` and Customized Comparison Functions

```cpp
struct P {
    int x, y;

    bool operator<(const P &p) const {
        if(x != p.x) {
            return x < p.x;
        } else {
            return y < p.y;
        }
    }
};

bool cmp(const P &a, const P &b) {
    if(a.y != b.y) {
        return a.y < b.y;
    } else {
        return a.x < b.x;
    }
}

int main() {
    vector<P> v;
    v.push_back({1, 2});
    v.push_back({2, 1});
    // small -> large
    sort(v.begin(), v.end());
    // {(1,2), (2,1)}
    sort(v.begin(), v.end(), cmp);
    // {(2,1), (1,2)}
    return 0;
}
```

2 Sorted, so what?

2.1 Binary Search - Sum of Two Values

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

int main() {
    int n, target;
    cin >> n >> target;
    vector<pair<int, int>> v(n);
    int cnt = 1;
    for(auto &i:v) {
        cin >> i.first;
        i.second = cnt++;
    }
    sort(begin(v), end(v));
    for(auto itr = begin(v); itr != end(v); itr++) {
        auto itr2 = lower_bound(begin(v), itr, make_pair(target - itr->first, 0));
        if(itr2 != itr and itr2->first + itr->first == target) {
            cout << itr->second << " " << itr2->second << endl;
            return 0;
        }
    }
    cout << "IMPOSSIBLE" << endl;
    return 0;
}
```

2.2 Stick Lengths

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

typedef long long ll;

int main() {
    ll n;
    cin >> n;
    vector<ll> v(n);
    for(auto &i:v)
        cin >> i;


    sort(begin(v), end(v));

    //find the median
    ll opt = v[n / 2];
    ll ans = 0;
    for(int i = 0; i < n; i++)
        ans += abs(v[i] - opt);
    cout << ans;

}
```

3 Just brute force it

3.1 for loop – UVa 725

```cpp
#include <bits/stdc++.h>
using namespace std;
void solve(int n) {
    bool found = false;
    for(int a = 0; a < 10; a++) {
        for(int b = 0; b < 10; b++) {
            for(int c = 0; c < 10; c++) {
                for(int d = 0; d < 10; d++) {
                    for(int e = 0; e < 10; e++) {
                        int numerator = a * 10000 + b * 1000 + c * 100 + d * 10 + e;
                        if(numerator % n == 0) {
                            int denominator = numerator / n;
                            string nu = to_string(numerator), de = to_string(denominator);
                            while(nu.size() < 5)nu = "0" + nu;
                            while(de.size() < 5)de = "0" + de;
                            set<char>s;
                            for(auto i:nu)s.insert(i);
                            for(auto i:de)s.insert(i);
                            if(s.size() == 10){
                                found = true;
                                cout<<nu<<" / "<<de<<" = "<<n<<endl;
                            }
                        }
                    }
                }
            }
        }
    }
    if(!found) cout << "There are no solutions for " << n << "." << endl;
}
int main() {
    int n; bool first = true;
    while(cin >> n) {
        if(n == 0) {
            return 0;
        } else {
            if(first) first = false;
            else cout << endl;
            solve(n);
        }
    }
}
```

3.2 Depth-First Search(DFS)

3.2.1 Generating Subset - Apple Division

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

typedef long long ll;

ll ans = LLONG_MAX, num[20], sum = 0;
int n;

void dfs(int step, ll sub_sum) {
    if(step == n) {
        ans = min(ans, abs(sub_sum - (sum - sub_sum)));
    } else {
        // choose
        dfs(step + 1, sub_sum + num[step]);

        // not choose
        dfs(step + 1, sub_sum + 0);
    }
}

int main() {
    cin >> n;
    for(int i = 0; i < n; i++) {
        cin >> num[i];
        sum += num[i];
    }
    dfs(0,0);
    cout<<ans<<endl;

    return 0;
}
```

3.2.2 Backtrack - Chessboard and Queens

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

const int n = 8;
string maze[n];
int pos[n];

int ans = 0;

void dfs(int row = 0) {
    if(row == n)
        ans++;
    else {
        for(pos[row] = 0; pos[row] < 8; pos[row]++) {
            bool ok = true;
            if(maze[row][pos[row]] == '*')ok = 0;
            for(int past = 0; past < row; past++) {
                if(pos[past] == pos[row])ok = false;
                if(pos[past] - pos[row] == past - row)ok = false;
                if(pos[row] - pos[past] == past - row)ok = false;
            }
            if(ok) {
                dfs(row + 1);
            }
        }
    }
}


int main() {
    for(int i = 0; i < n; i++) {
        cin >> maze[i];
    }
    dfs();
    cout << ans << endl;
    return 0;
}
```

3.2.3 Meet in the middle – Codeforces 888E

```cpp
#include <bits/stdc++.h>
using namespace std;
int n, m;
void dfs(vector<int> &num, vector<int> &result, int sum = 0, int cur = 0) {
    if(cur == num.size()) {
        result.push_back(sum);
    } else {
        // not choose
        dfs(num, result, (sum + num[cur]) % m, cur + 1);

        // choose
        dfs(num, result, sum, cur + 1);
    }
}
int main() {
    cin >> n >> m;
    vector<int> result[2], num[2];
    for(int i = 0; i < n; i++) {
        int v;
        cin >> v;
        if(i % 2 == 0) {
            num[0].push_back(v);
        } else {
            num[1].push_back(v);
        }
    }
    for(int i = 0; i < 2; i++) {
        dfs(num[i], result[i]);
        sort(begin(result[i]), end(result[i]));
    }
    int ans = 0;
    for(auto a: result[0]) {
        auto iterator = lower_bound(begin(result[1]), end(result[1]), m - a);
        if(iterator != begin(result[1])) {
            --iterator;
            ans = max(ans, a + (*iterator));
        }
        ans = max(ans, (a + result[1].back()) % m);
    }
    cout << ans << endl;
    return 0;
}
```

3.3 Bitmask

3.3.1 Generating Subset - Apple Division

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

typedef long long ll;

ll ans = LLONG_MAX, num[20], sum = 0;
int n;


int main() {
    cin >> n;
    for(int i = 0; i < n; i++) {
        cin >> num[i];
        sum += num[i];
    }
    for(int mask = 0; mask < (1 << n); mask++) {
        ll sub_sum = 0;
        for(int bit = 0; bit < n; bit++) {
            if((1 << bit) & mask)
                sub_sum += num[bit];
        }
        ans = min(ans, abs(sub_sum - (sum - sub_sum)));
    }


    cout << ans << endl;

    return 0;
}
```