Homework of algorithm analysis

软工一班 成子谦 201730681303

Chapter. 4

We can solve this program with divide and conquer.

Let team number as 4, we can write it into a $1 * 4$ matrix.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \tag{1}$$

The situation of the first day is:

$$\begin{bmatrix} 2 & 1 & 4 & 3 \end{bmatrix} \tag{2}$$

The situation of day 2 and day 3 likes:

$$\begin{bmatrix} 3 & 4 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix} \tag{3}$$

We can find the rules by combining the matrix (1), (2), (3) together:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix} \tag{4}$$

In this way, we can construct a matrix for the situation of n teams easily:

$$\begin{bmatrix} 1 & 2 & \cdots & n-1 & n \\ 2 & 1 & \cdots & n & n-1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ n-1 & n & \cdots & 1 & 2 \\ n & n-1 & \cdots & 2 & 1 \end{bmatrix} \tag{5}$$

So the algorithm is:

- if current n (current size of matrix) is not equal to 2, divide the matrix into four parts (up-left, down-left, up-right, down-right) and deal with up-left part and up-right part.
- if current n equals to 2, let the down-right element equals to up-left element and down-left element equals to up-right element.
- Since it's a procedure of finishing table, the time complexity of this algorithm is $O(n^2)$.

Source code:

```
#include <iostream>

using namespace std;

const int maxn = 1024 + 10;
long long n = 1, k;

void output(int a[][maxn]) {
    for (int i = 1; i <= n; i++) {
        if (i == 1) cout << "Ball team: ";
        else cout << "    Day " << i - 1 << ": ";
        for (int j = 1; j <= n; j++) cout << a[i][j] << " ";
        cout << endl;
    }
}

void solve(int a[][maxn], long long n, int x, int y) {
    if (n == 2) {
        a[x + 1][y + 1] = a[x][y];
        a[x + 1][y] = a[x][y + 1];
        return;
    }
    // divide
    solve(a, n / 2, x, y); solve(a, n / 2, x, y + n / 2);
    // conquer
    for (int i = x; i < x + n / 2; i++)
        for (int j = y; j < y + n / 2; j++) {
            a[i + n / 2][j + n / 2] = a[i][j];
        }
    for (int i = x + n / 2; i < x + n; i++)
        for (int j = y; j < y + n / 2; j++) {
            a[i][j] = a[i - n / 2][j + n / 2];
        }
}

int main() {
    cout << "The k is: ";
    cin >> k;
    for (int i = 1; i <= k; i++) n <<= 1;
    cout << "The n is: " << n << endl;
    int a[n + 1][maxn];
    for (int i = 1; i <= n; i++) a[1][i] = i;
    solve(a, n, 1, 1);
    output(a);
    return 0;
}
```

The result is:

```
The k is: 3
The n is: 8
Ball team: 1 2 3 4 5 6 7 8
    Day 1: 2 1 4 3 6 5 8 7
    Day 2: 3 4 1 2 7 8 5 6
    Day 3: 4 3 2 1 8 7 6 5
    Day 4: 5 6 7 8 1 2 3 4
    Day 5: 6 5 8 7 2 1 4 3
    Day 6: 7 8 5 6 3 4 1 2
    Day 7: 8 7 6 5 4 3 2 1
```

Chapter. 5

Just solve problem with topological sort, dfs and bfs.

Source code:

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <queue>
#include <set>

using namespace std;

vector<char>g[26];
int vis[26], inDeg[26];
set<char>point;

void init() {
    for (int i = 0; i < 26; i++) {
        g[i].clear();
        vis[i] = inDeg[i] = 0;
    }
    point.clear();
    for (int i = 0; i < 6; i++) point.insert(char('a' + i));
}

void buildDirectedGraph() {
    init();
    g['a' - 'a'].emplace_back('b'); inDeg['b' - 'a']++;
    g['a' - 'a'].emplace_back('c'); inDeg['c' - 'a']++;
    g['b' - 'a'].emplace_back('d'); inDeg['d' - 'a']++;
    g['c' - 'a'].emplace_back('e'); inDeg['e' - 'a']++;
    g['c' - 'a'].emplace_back('f'); inDeg['f' - 'a']++;
    g['d' - 'a'].emplace_back('c'); inDeg['c' - 'a']++;
    g['f' - 'a'].emplace_back('e'); inDeg['e' - 'a']++;
    for (int i = 0; i < 26; i++) {
        if (!g[i].empty())
            sort(g[i].begin(), g[i].end());
    }
}

void buildUndirectedGraph() {
    init();
    g['a' - 'a'].emplace_back('b'); g['b' - 'a'].emplace_back('a');
    g['a' - 'a'].emplace_back('c'); g['c' - 'a'].emplace_back('a');
    g['b' - 'a'].emplace_back('d'); g['d' - 'a'].emplace_back('b');
    g['c' - 'a'].emplace_back('e'); g['e' - 'a'].emplace_back('c');
    g['c' - 'a'].emplace_back('f'); g['f' - 'a'].emplace_back('c');
    g['d' - 'a'].emplace_back('c'); g['c' - 'a'].emplace_back('d');
    g['f' - 'a'].emplace_back('e'); g['e' - 'a'].emplace_back('f');
    for (int i = 0; i < 26; i++)
        if (!g[i].empty())
            sort(g[i].begin(), g[i].end());
}

int findZeroDegreePoint() {
    for (int i = 0; i < 26; i++)
        if (point.count(char('a' + i)) && !vis[i] && !inDeg[i])
            return i;
    return -1;
```

```cpp
}

void topologicalSort() {
    while (1) {
        int curr = findZeroDegreePoint();
        if (curr == -1) break;
        cout << char('a' + curr) << " ";
        vis[curr] = 1;
        for (auto i : g[curr]) inDeg[i - 'a']--;
    }
}

void dfs(char curr) {
    cout << curr << " ";
    vis[curr - 'a'] = 1;
    for (auto i : g[curr - 'a']) {
        if (!vis[i - 'a']) dfs(i);
    }
}

int main() {
    // topological sort
    buildDirectedGraph();
    cout << "The topological sort sequence is: ";
    topologicalSort();
    cout << endl;

    // dfs
    buildUndirectedGraph();
    cout << "The DFS sequence is: ";
    for (int i = 0; i < 26; i++) {
        if (!g[i].empty()) {
            dfs(char('a' + i));
            break;
        }
    }
    cout << endl;

    // bfs
    buildUndirectedGraph();
    cout << "The BFS sequence is: ";
    queue<char>q;
    while (!q.empty()) q.pop();
    for (int i = 0; i < 26; i++) {
        if (!g[i].empty()) {
            q.push(char('a' + i));
            break;
        }
    }
    while (!q.empty()) {
        char curr = q.front();
        q.pop();
        if (vis[curr - 'a']) continue;
        cout << curr << " ";
        vis[curr - 'a'] = 1;
        for (auto i : g[curr - 'a']) q.push(i);
    }
    cout << endl;
    return 0;
}
```

The result is:

```
The topological sort sequence is: a b d c f e
The DFS sequence is: a b d c e f
The BFS sequence is: a b c d e f
```