递归

什么是递归

- 递归是一种思维方式

- 我们平常在做事情的时候往往发现一件事情可以被分成若干件事情，做完这若干件事情之后这件事情就解决了，但可能这若干见事情还可以被分成一些小事情，所以我们在一开始就要分好工，将所有的最小的事情一件件的自底向上的做过去。

- 递归的思维相当于你有了很多帮手，你是最终的BOSS，你把一件事情分成若干件事情后，每件事情都由你的帮手帮你解决，你只需要负责他们解决完成后，将他们解决问题的方案进行一些整理与合并得到你当前这个问题的解决方案。

- 在计算机中，这些帮手就是内存，递归调用消耗的是栈内存

归并排序

- 首先，假如著名的计算机科学家Dijkstra当年没有把递归加入到计算机里面，我们来做做归并排序这个题目。

- 8 7 6 5 4 3 2 1

- 假如我们需要将这8个数按照从小到大排序，按照归并排序的思想，假设计算机无法表达递归，写出来的代码会是什么样子的呢?

代码实现

```
void merge_sort(){
    for (int len = 1; len < n; len *= 2) {
        for (int i = 0; i + len < n; i += 2 * len) {
            merge(i, i + len, len);
        }
    }
    for (int i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
}
```

- 我们再来看一下递归版本的归并排序。

```
void merge_sort(int l, int r)
{
    int m = (l + r) / 2;
    merge_sort(l, m); merge_sort(m + 1, r);
    merge(l, l + m, r);
}
```

- 解决左边的排序，解决右边的排序，合并左右的排序，就结束了
- 编程人员只需要专注于解决当前的问题，无需在细节上折腾太久
- 因此递归，不仅仅是一个技能点，它更是一种思维方式。

输出全排列

```
void dfs(int dep) {
    if (dep == N) {
        for (int i = 0; i < N; i++) {
            cout << rec[i] << " ";
        }
        cout << endl;
        return ;
    }
    for (int i = 1; i <= N; i++) if(!vis[i]) {
        rec[dep] = i;
        vis[i] = true;
        dfs(dep + 1);
        vis[i] = false;
    }
}
```

走迷宫

```
bool vis[11][11];
int cnt;
int dx[] = {1,-1,0,0};
int dy[] = {0, 0,1,-1};
void dfs(int x, int y) {
    if (x == n - 1 && y == m - 1) {
        cnt++;
        return ;
    }
    for (int i = 0; i < 4; i++) {
        int tx = x + dx[i];
        int ty = y + dy[i];
        if (tx < 0 || tx >= n || ty < 0 || ty >= m || a[tx][ty]=='#')  {
            continue;
        }
        if (vis[tx][ty]) {
            continue;
        }
        vis[tx][ty] = true;
        dfs(tx, ty);
        vis[tx][ty] = false;
    }
}
```

二叉树遍历

```c
void dfs(int u) {
    printf("%d ", u);
    if (ch[u][0] != -1)
        dfs(ch[u][0]);
    if (ch[u][1] != -1)
        dfs(ch[u][1]);
}

void dfs(int u) {
    if (ch[u][0] != -1)
        dfs(ch[u][0]);
    printf("%d ", u);
    if (ch[u][1] != -1)
        dfs(ch[u][1]);
}

void dfs(int u) {
    if (ch[u][0] != -1)
        dfs(ch[u][0]);
    if (ch[u][1] != -1)
        dfs(ch[u][1]);
    printf("%d ", u);
}
```

表达式计算

- 给你一个四则运算的表达式，包含括号，数字，四则运算符，求表达式的值

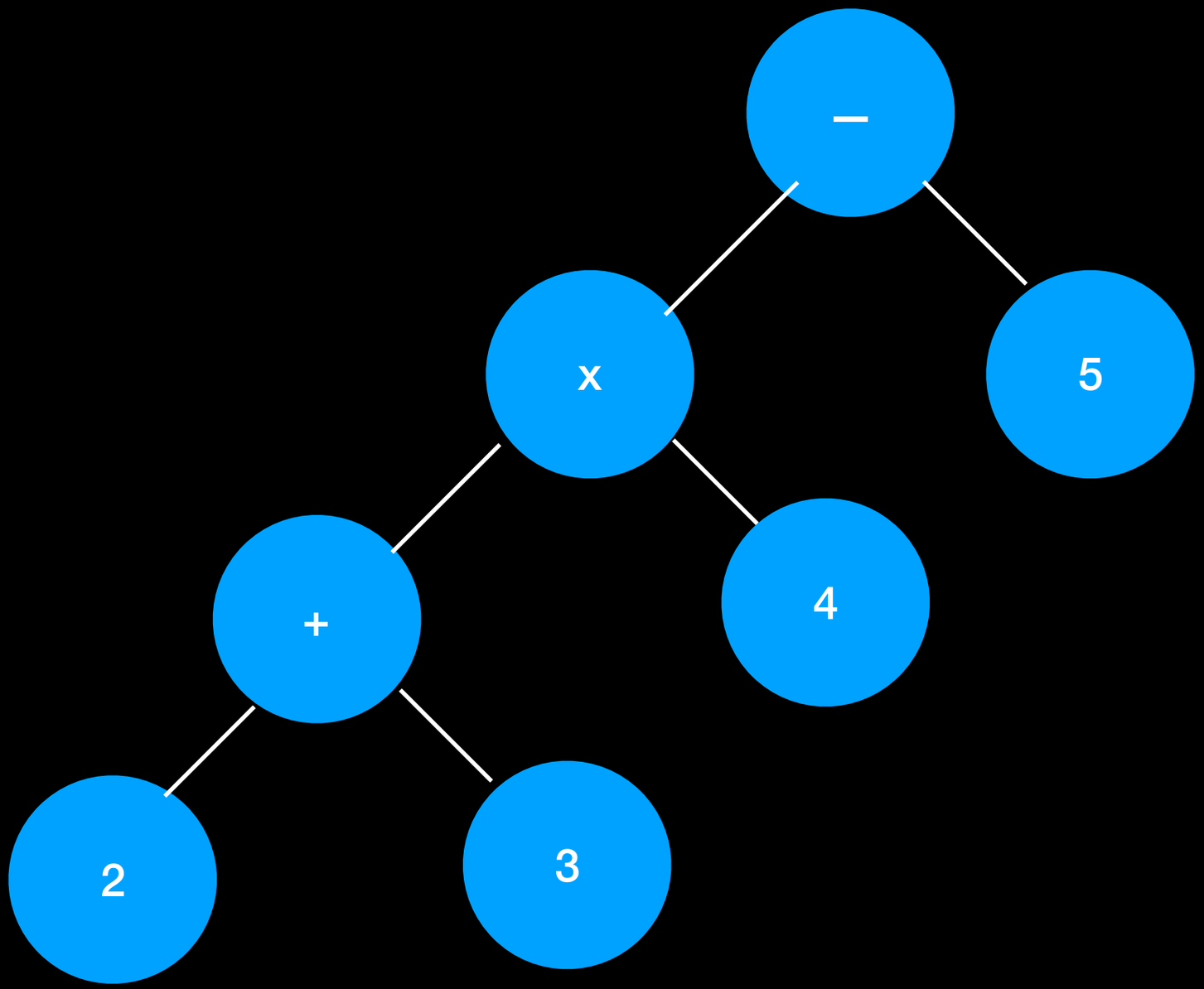
- 比如(2+3)*4 - 5， 我们发现减号是最后被运算的，于是我们可以将减号左右的式子分开算，最后相减得到答案

- 这提示我们问题可以递归解决

- 关键是要找到表达式中最后被计算的运算符

- 括号里面的优先与括号外面的，乘除优先与加减

- 递归的过程本质上会形成一棵表达式树


−
x
5
+
4
2
3

- 我们只需要对这个表达式树进行后序遍历就可以计算表达式的值