

# Standard Code Library

Your TeamName

Your School

December 16, 2021

# Contents

<b>编译原理实验 1</b>	<b>2</b>
代码 . . . . .	2
<b>编译原理实验 2</b>	<b>5</b>
代码 . . . . .	5
<b>编译原理实验 3</b>	<b>12</b>
代码 . . . . .	12
<b>数据结构</b>	<b>15</b>
ST 表 . . . . .	15
<b>数学</b>	<b>15</b>
类欧几里得 . . . . .	15
<b>图论</b>	<b>15</b>
LCA . . . . .	15
<b>计算几何</b>	<b>16</b>
二维几何: 点与向量 . . . . .	16
<b>字符串</b>	<b>17</b>
后缀自动机 . . . . .	17
<b>杂项</b>	<b>17</b>
STL . . . . .	17

## 编译原理实验 1

### 代码

- 需要 C++11

```
1  #include<bits/stdc++.h>
2
3  using namespace std;
4
5  enum {
6      Comment = 0, Main, If, Then, While, Do, Static, Int, Double, Struct,
7      Break, Else, Long, Switch, Case, Typedef, Char, Return, Const, Float,
8      Short, Continue, For, Void, Sizeof, ID, NUM, ADD, SUB, MUL,
9      DIV, COLON, DEFINE, LT, NE, LE, GT, GE, EQ, Default,
10     Do1, SEMI, LB, RB
11 };
12
13 char name[][10] = {
14     "Comment", "Main", "If", "Then", "While", "Do", "Static", "Int", "Double", "Struct",
15     "Break", "Else", "Long", "Switch", "Case", "Typedef", "Char", "Return", "Const", "Float",
16     "Short", "Continue", "For", "Void", "Sizeof", "ID", "NUM", "ADD", "SUB", "MUL",
17     "DIV", "COLON", "DEFINE", "LT", "NE", "LE", "GT", "GE", "EQ", "Default",
18     "Do1", "SEMI", "LB", "RB"
19 };
20
21
22 struct Token {
23     int syn = 0;
24     string token;
25     int val = 0;
26
27     void print() const {
28         if (syn == -2) {
29             printf("<%d,%s>\n", syn, token.c_str());
30             return;
31         }
32         if (syn == NUM) {
33             printf("<%d,", syn);
34             for (int i = 31; i >= 0; --i) {
35                 printf("%d", (val >> i) & 1);
36             }
37             printf("b>\n");
38         }
39         // else if (syn == ID)
40         //     printf("<%d,%s>\n", syn, token.c_str());
41         else
42             printf("<%d,%s>\n", syn, token.c_str());
43     }
44 };
45
46
47 int identifierNum = RB + 1;
48 unordered_map<string, int> identifier;
49
50 void initIdentifier() {
51     identifier["#"] = Comment;
52     identifier["main"] = Main;
53     identifier["if"] = If;
54     identifier["then"] = Then;
55     identifier["while"] = While;
56     identifier["do"] = Do;
57     identifier["static"] = Static;
58     identifier["int"] = Int;
59     identifier["double"] = Double;
60     identifier["struct"] = Struct;
61     identifier["break"] = Break;
62     identifier["else"] = Else;
63     identifier["long"] = Long;
64     identifier["switch"] = Switch;
65     identifier["case"] = Case;
```

```

66     identfier["typedef"] = Typedef;
67     identfier["char"] = Char;
68     identfier["return"] = Return;
69     identfier["const"] = Const;
70     identfier["float"] = Float;
71     identfier["short"] = Short;
72     identfier["continue"] = Continue;
73     identfier["for"] = For;
74     identfier["void"] = Void;
75     identfier["sizeof"] = Sizeof;
76     identfier["default"] = Default;
77
78 }
79
80 char *p, *lastp;
81 int line = 0;
82
83
84 Token *next() {
85     auto *tk = new Token;
86     tk->token = *p;
87     while (*p) { //判断字符串是否结束
88         if (*p == '\n') {
89             printf("%d: ", line, (int) (p - lastp), lastp);
90             while (lastp < p) putchar(*(lastp++));
91             printf("\n\n");
92             ++p;
93             lastp = p;
94             line = line + 1;
95         } else if (isalpha(*p)) {
96             tk->syn = ID;
97             tk->token = "";
98             while (isalpha(*p) || isdigit(*p)) {
99                 tk->token += *p;
100                 ++p;
101             }
102             if (identfier[tk->token] != 0) {
103                 if (identfier[tk->token] <= RB) {
104                     tk->syn = identfier[tk->token];
105                 } else {
106                     tk->val = identfier[tk->token];
107                 }
108             } else {
109                 tk->val = identfier[tk->token] = ++identfierNum;
110             }
111             return tk;
112         } else if (*p >= '0' && *p <= '9') {
113             tk->syn = NUM;
114             tk->val = 0;
115             tk->token = "";
116             while (*p >= '0' && *p <= '9' || isalpha(*p)) {
117                 if (isalpha(*p)) {
118                     tk->syn = -2;
119                 } else {
120                     tk->val = tk->val * 10 + *p - '0';
121                 }
122                 tk->token += *p;
123                 ++p;
124             }
125             return tk;
126         } else {
127             tk->token = *p;
128             if (*p == '+') {
129                 ++p;
130                 tk->syn = ADD;
131                 return tk;
132             } else if (*p == '-') {
133                 ++p;
134                 tk->syn = SUB;
135                 return tk;
136             } else if (*p == '*') {

```

```

137     ++p;
138     tk->syn = MUL;
139     return tk;
140 } else if (*p == '/') {
141     ++p;
142     if (*p == '*') {
143         while (true) {
144             ++p;
145             if (!*p) {
146                 tk->syn = -1;
147                 return tk;
148             }
149             while (*p && *p != '*')
150                 ++p;
151             if (*p == '*' && *(p + 1) == '/') {
152                 p += 2;
153                 break;
154             }
155         }
156         continue;
157     } else {
158         tk->syn = DIV;
159         return tk;
160     }
161 } else if (*p == ':') {
162     ++p;
163     if (*p == '=') {
164         ++p;
165         tk->token += *p;
166         tk->syn = DEFINE;
167     } else {
168         tk->syn = COLON;
169     }
170     return tk;
171 } else if (*p == '<') {
172     ++p;
173     if (*p == '>') {
174         ++p;
175         tk->token += *p;
176         tk->syn = NE;
177     } else if (*p == '=') {
178         ++p;
179         tk->token += *p;
180         tk->syn = LE;
181     } else {
182         tk->syn = LT;
183     }
184     return tk;
185 } else if (*p == '>') {
186     ++p;
187     if (*p == '=') {
188         ++p;
189         tk->token += *p;
190         tk->syn = GE;
191     } else {
192         tk->syn = GT;
193     }
194     return tk;
195 } else if (*p == '=') {
196     ++p;
197     tk->syn = EQ;
198     return tk;
199 } else if (*p == ';') {
200     ++p;
201     tk->syn = SEMI;
202     return tk;
203 } else if (*p == '(') {
204     ++p;
205     tk->syn = LB;
206     return tk;
207 } else if (*p == ')') {

```

```

208         ++p;
209         tk->syn = RB;
210         return tk;
211     } else {
212         if (!isblank(*p)) {
213             tk->syn = -2;
214             tk->token = "";
215             if (*p && *p != '\n' && !isblank(*p)) {
216                 tk->token += *p;
217                 ++p;
218             }
219             return tk;
220         }
221         ++p;
222     }
223 }
224 }
225 printf("%d: ", line);
226 while (lastp < p) putchar(*(lastp++));
227 tk->syn = -1;
228 return tk;
229 }
230
231 char s[1000000];
232
233 int main() {
234     initIdentifier();
235     FILE *f = fopen("1.txt", "r");
236     fread(s, 1000000, 1000000, f);
237     p = lastp = s;
238     Token *tk = next();
239     while (tk->syn != -1) {
240         tk->print();
241         tk = next();
242     }
243     return 0;
244 }

```

## 编译原理实验 2

### 代码

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  typedef string grammarElement;
5  typedef pair<char, char> pcc;
6  map<char, vector<grammarElement>> grammarSet, grammarTmp;
7  map<char, int> flag;
8  map<char, set<char>> firstSet;
9  map<char, set<char>> followSet;
10 map<pcc, set<string>> M;
11 char S;
12 int err = 0;
13
14 void dfs(char ch)
15 {
16     // cout << "dfs" << ch << "\n";
17     flag[ch] = 2; // working
18     for (auto str : grammarSet[ch])
19     {
20         if (isupper(str[0]))
21         {
22             if (flag[str[0]] == 1)
23             {
24                 for (auto i : firstSet[str[0]])
25                 {
26                     if (i != '%')
27                     {
28                         firstSet[ch].insert(i);

```

```

29         }
30     }
31 }
32 else if (flag[str[0]] == 0)
33 {
34     dfs(str[0]);
35     for (auto i : firstSet[str[0]])
36     {
37         if (i != '%')
38         {
39             firstSet[ch].insert(i);
40         }
41     }
42 }
43 else
44 {
45     puts("error: exist left recursion");
46     err = 1;
47     return;
48 }
49 }
50 else
51 {
52     firstSet[ch].insert(str[0]);
53 }
54 }
55 flag[ch] = 1; // finish
56 // cout << "outdfs" << ch << "\n";
57 }
58
59 char nm[10000];
60 int e = 0;
61 vector<int> eg[10000];
62
63 void dfs2(string perfix, int x, bool isend)
64 {
65     cout << perfix;
66     cout << (isend ? "└─" : "└─");
67     cout << nm[x] << endl;
68     ;
69     for (int i = 0; i < eg[x].size(); ++i)
70         dfs2(perfix + (isend ? " " : "└─"), eg[x][i], i == eg[x].size() - 1);
71 }
72
73 char nt = 'H';
74
75 int main()
76 {
77     freopen("llltest.in", "r", stdin);
78     int n;
79     cin >> n;
80     for (int i = 0; i < n; ++i)
81     {
82         string x, y;
83         cin >> x >> y;
84         if (x.length() > 1 || !isupper(x[0]))
85         {
86             puts("input error: x must be in [A-Z]");
87             --i;
88             continue;
89         }
90         if (!grammarSet[x[0]].empty())
91         {
92             puts("warning: redefining");
93         }
94         if (i == 0)
95         {
96             S = x[0];
97         }
98         err = 0;
99         string tmp = "";

```

```

100     for (auto ch : y)
101     {
102         if (ch == '|' || ch == '#')
103         {
104             if (ch == '#')
105             {
106                 err = 1;
107                 grammarSet[x[0]].clear();
108                 break;
109             }
110             grammarSet[x[0]].push_back(tmp);
111             tmp = "";
112         }
113         else
114         {
115             tmp += ch;
116         }
117     }
118     grammarSet[x[0]].push_back(tmp);
119     if (err)
120     {
121         puts("error: do not use #");
122         --i;
123         continue;
124     }
125 }
126
127 for (auto &i : grammarSet)
128 {
129
130     for (const auto &j : grammarSet)
131     {
132         if (j.first == i.first)
133         {
134             int flag = 0;
135             for (int k = 0; k < i.second.size(); ++k)
136             {
137                 if (i.second[k][0] == i.first)
138                 {
139                     flag = 1;
140                     break;
141                 }
142             }
143
144             if (!flag)
145                 continue;
146             cout << "let " << nt << " as " << i.first << "" << endl;
147             auto tmp = i.second;
148             i.second.clear();
149             for (auto k : tmp)
150             {
151                 if (k[0] != i.first)
152                 {
153                     i.second.push_back(k + nt);
154                 }
155             }
156             grammarTmp[nt].push_back("");
157             for (auto k : tmp)
158             {
159                 if (k[0] == i.first)
160                 {
161                     grammarTmp[nt].push_back(k.substr(1) + nt);
162                 }
163             }
164             ++nt;
165         }
166         else if (j.first < i.first)
167         {
168             for (int k = 0; k < i.second.size(); )
169             {
170                 if (i.second[k][0] == j.first)

```



```

171         {
172             for (auto t : j.second)
173                 i.second.push_back(t + i.second[k].substr(1));
174             i.second.erase(i.second.begin() + k);
175         }
176         else
177         {
178             ++k;
179         }
180     }
181     int flag = 0;
182     for (int k = 0; k < i.second.size(); ++k)
183     {
184         if (i.second[k][0] == i.first)
185         {
186             flag = 1;
187             break;
188         }
189     }
190
191     if (!flag)
192         continue;
193     cout << "let " << nt << " as " << i.first << " " << endl;
194     auto tmp = i.second;
195     i.second.clear();
196     for (auto k : tmp)
197     {
198         if (k[0] != i.first)
199         {
200             i.second.push_back(k + nt);
201         }
202     }
203     grammarTmp[nt].push_back("%");
204     for (auto k : tmp)
205     {
206         if (k[0] == i.first)
207         {
208             grammarTmp[nt].push_back(k.substr(1) + nt);
209         }
210     }
211     ++nt;
212 }
213 }
214 }
215 for (auto i : grammarTmp)
216 {
217     grammarSet[i.first] = i.second;
218 }
219 for (auto i : grammarSet)
220 {
221     cout << i.first << "->";
222     for (auto j : i.second)
223     {
224         cout << j << "|";
225     }
226     cout << endl;
227 }
228
229 for (auto x : grammarSet)
230 {
231     // cout<<x.first<<endl;
232     // for (auto i : grammarSet[x.first])
233     // {
234     //     cout << x.first << "->" << i << "\n";
235     // }
236     if (!flag[x.first])
237     {
238         dfs(x.first);
239     }
240 }
241 for (auto x : firstSet)

```

```

242 {
243     cout << "first[" << x.first << "]={";
244     for (auto i : x.second)
245     {
246         cout << i << ",";
247     }
248     cout << "]\n";
249 }
250 followSet[S].insert('#');
251 int cflag = 1;
252 while (cflag)
253 {
254     cflag = 0;
255     for (auto x : grammarSet)
256     {
257         for (auto str : x.second)
258         {
259             for (int i = 0; i < str.size(); ++i)
260             {
261                 if (isupper(str[i]))
262                 {
263                     int size = followSet[str[i]].size();
264                     if (i + 1 < str.size())
265                     {
266                         if (isupper(str[i + 1]))
267                         {
268                             for (auto j : firstSet[str[i + 1]])
269                             {
270                                 if (j != '%')
271                                 {
272                                     followSet[str[i]].insert(j);
273                                 }
274                             }
275                             if (firstSet[str[i + 1]].find('%') != firstSet[str[i + 1]].end())
276                             {
277                                 for (auto j : followSet[str[i + 1]])
278                                 {
279                                     if (j != '%')
280                                     {
281                                         followSet[str[i]].insert(j);
282                                     }
283                                 }
284                             }
285                         }
286                     }
287                     else
288                     {
289                         if (str[i + 1] != '%')
290                         {
291                             followSet[str[i]].insert(str[i + 1]);
292                         }
293                     }
294                 }
295             }
296             else
297             {
298                 for (auto j : followSet[x.first])
299                 {
300                     if (j != '%')
301                     {
302                         followSet[str[i]].insert(j);
303                     }
304                 }
305                 if (followSet[str[i]].size() > size)
306                 {
307                     cflag = 1;
308                 }
309             }
310         }
311     }
312 }

```

```

313     }
314     for (auto x : followSet)
315     {
316         if (!isupper(x.first))
317         {
318             continue;
319         }
320         cout << "follow[" << x.first << "]={";
321         for (auto i : x.second)
322         {
323             cout << i << ",";
324         }
325         cout << "]\n";
326     }
327
328     for (auto x : grammarSet)
329     {
330         for (auto str : x.second)
331         {
332             if (isupper(str[0]))
333             {
334                 for (auto i : firstSet[str[0]])
335                 {
336                     if (!isupper(i) && i != '%')
337                     {
338                         M[pcc(x.first, i)].insert(str);
339                     }
340                 }
341             }
342             else if (str[0] != '%')
343             {
344                 M[pcc(x.first, str[0])].insert(str);
345             }
346             if (str[0] == '%' || firstSet[str[0]].find('%') != firstSet[str[0]].end())
347             {
348                 for (auto i : followSet[x.first])
349                 {
350                     M[pcc(x.first, i)].insert("%");
351                 }
352             }
353             if (firstSet[x.first].find('%') != firstSet[x.first].end() && followSet[x.first].find('#') !=
⇒ followSet[x.first].end())
354             {
355                 M[pcc(x.first, '#')].insert("%");
356             }
357         }
358     }
359     set<char> row, col;
360     for (auto i : M)
361     {
362         row.insert(i.first.first);
363         col.insert(i.first.second);
364     }
365     cout << setiosflags(ios::left);
366     cout << setw(10) << "";
367     for (auto i : col)
368     {
369         cout << setw(10) << i;
370     }
371     cout << "\n";
372     for (auto i : row)
373     {
374         cout << setw(10) << i;
375         for (auto j : col)
376         {
377             string tmp;
378             for (auto k : M[pcc(i, j)])
379             {
380                 if (k.back() == '%' && M[pcc(i, j)].size() > 1)
381                 {
382                     continue;

```

```

383         }
384         tmp += k + ";";
385     }
386     cout << setw(10) << (tmp.size() ? i + (">" + tmp) : "");
387 }
388 cout << "\n";
389 }
390 string now;
391 cin >> now;
392 {
393     now += "#";
394     int it = 0;
395     stack<char> stk;
396     stk.push(0);
397     stk.push(1);
398     int flag = 1;
399     nm[0] = '#';
400     nm[++e] = S;
401
402     while (flag && stk.size())
403     {
404         auto tmp = stk;
405         string out = "";
406         int xx = stk.top();
407         char x = nm[stk.top()];
408         stk.pop();
409         while (!tmp.empty())
410         {
411             out = nm[tmp.top()] + (out);
412             tmp.pop();
413         }
414         cout << setw(30) << out << "\t" << now.substr(it) << endl;
415         if (!isupper(x))
416         {
417             if (it < now.length() && x == now[it])
418             {
419                 if (it + 1 < now.length())
420                     ++it;
421             }
422             else
423             {
424                 puts("error 1");
425                 return 0;
426             }
427         }
428         else if (x == '#')
429         {
430             if (now[it] == '#')
431             {
432                 flag = false;
433             }
434             else
435             {
436                 puts("error 2");
437                 return 0;
438             }
439         }
440         else if (M[pcc(x, now[it])].size() != 0)
441         {
442             for (auto i : M[pcc(x, now[it])])
443             {
444                 if (i != "%")
445                 {
446                     for (int j = i.length() - 1; j >= 0; --j)
447                     {
448                         stk.push(++e);
449                         nm[e] = i[j];
450                         eg[xx].push_back(e);
451                     }
452                 }
453             }

```

```

454         }
455         else
456         {
457             puts("error 3");
458             return 0;
459         }
460     }
461     puts("success");
462 }
463 dfs2(" ", 1, 1);
464 return 0;
465 }

```

## 编译原理实验 3

### 代码

```

1  class Rule(object):
2      def __init__(self):
3          self.left = ""
4          self.right = []
5
6  VT = []
7  VN = []
8  Rules = []
9  FirstVT = []
10 LastVT = []
11 rule_list = []
12 OG = []
13 og_stack = []
14
15 def create_rule_list():
16     for i in range(0, len(Rules)):
17         Rules[i] = Rules[i].replace(' ', '')
18         rule = Rule()
19         rule_list.append(rule)
20     for j in range(0, len(Rules)):
21         arrow_pos = Rules[j].find('-')
22         rule_list[j].left = Rules[j][0:arrow_pos]
23         rule_list[j].right = list(Rules[j][arrow_pos + 2:])
24         while "" in rule_list[j].right:
25             pos = rule_list[j].right.index("")
26             new_sym = "".join(rule_list[j].right[pos - 1: pos + 1])
27             del rule_list[j].right[pos]
28             del rule_list[j].right[pos - 1]
29             if new_sym not in rule_list[j].right:
30                 rule_list[j].right.append(new_sym)
31
32 def identify_vt_and_vn():
33     for i in range(0, len(rule_list)):
34         if rule_list[i].left not in VN:
35             VN.append(rule_list[i].left)
36         for j in range(len(rule_list[i].right)):
37             if rule_list[i].right[j].isupper():
38                 if rule_list[i].right[j] not in VN:
39                     VN.append(rule_list[i].right[j])
40             elif rule_list[i].right[j] != 'ε' and "" not in rule_list[i].right[j]:
41                 if rule_list[i].right[j] not in VT:
42                     VT.append(rule_list[i].right[j])
43             elif "" in rule_list[i].right[j]:
44                 if rule_list[i].right[j] not in VN:
45                     VN.append(rule_list[i].right[j])
46     VT.append('#')
47
48 def gen_firstvt(ch):
49     for i in range(len(rule_list)):
50         if rule_list[i].left == ch:
51             # 形如 U -> b... 之类的规则, 将 b 加入 U 的 FirstVT 集
52             if rule_list[i].right[0] in VT:
53                 if rule_list[i].right[0] not in FirstVT[VN.index(ch)]:

```

```

54         FirstVT[VN.index(ch)].append(rule_list[i].right[0])
55         # 形如  $U \rightarrow Vb\dots$  之类的规则, 将  $b$  加入  $U$  的  $FirstVT$  集
56         elif len(rule_list[i].right) > 1 and rule_list[i].right[1] in VT:
57             if rule_list[i].right[1] not in FirstVT[VN.index(ch)]:
58                 FirstVT[VN.index(ch)].append(rule_list[i].right[1])
59         # 形如  $U \rightarrow V\dots$  的规则, 将  $V$  的  $FirstVT$  集里的元素加入  $U$  的  $FirstVT$  集
60         if rule_list[i].right[0] in VN:
61             if not FirstVT[VN.index(rule_list[i].right[0])]:
62                 gen_firstvt(rule_list[i].right[0])
63             for c in FirstVT[VN.index(rule_list[i].right[0])]:
64                 if c not in FirstVT[VN.index(ch)]:
65                     FirstVT[VN.index(ch)].append(c)
66
67     def gen_lastvt(ch):
68         for i in range(len(rule_list)):
69             if rule_list[i].left == ch:
70                 if rule_list[i].right[-1] in VT:
71                     if rule_list[i].right[-1] not in LastVT[VN.index(ch)]:
72                         LastVT[VN.index(ch)].append(rule_list[i].right[-1])
73                 elif len(rule_list[i].right) > 1 and rule_list[i].right[-2] in VT and rule_list[i].right[-1] in VN:
74                     if rule_list[i].right[-2] not in LastVT[VN.index(ch)]:
75                         LastVT[VN.index(ch)].append(rule_list[i].right[-2])
76                 if rule_list[i].right[-1] in VN:
77                     if not LastVT[VN.index(rule_list[i].right[-1])]:
78                         gen_lastvt(rule_list[i].right[-1])
79                     for c in LastVT[VN.index(rule_list[i].right[-1])]:
80                         if c not in LastVT[VN.index(ch)]:
81                             LastVT[VN.index(ch)].append(c)
82
83     def create_og():
84         for i in range(len(rule_list)):
85             for j in range(0, len(rule_list[i].right) - 1):
86                 if rule_list[i].right[j] in VT and rule_list[i].right[j+1] in VT:
87                     OG[VT.index(rule_list[i].right[j])][VT.index(rule_list[i].right[j+1])] = '='
88                 if j < len(rule_list[i].right) - 2 and rule_list[i].right[j] in VT and rule_list[i].right[j+2] in VT:
89                     OG[VT.index(rule_list[i].right[j])][VT.index(rule_list[i].right[j+2])] = '='
90                 if rule_list[i].right[j] in VT and rule_list[i].right[j+1] in VN:
91                     for c in FirstVT[VN.index(rule_list[i].right[j+1])]:
92                         OG[VT.index(rule_list[i].right[j])][VT.index(c)] = '<'
93                 if rule_list[i].right[j] in VN and rule_list[i].right[j+1] in VT:
94                     for c in LastVT[VN.index(rule_list[i].right[j])]:
95                         OG[VT.index(c)][VT.index(rule_list[i].right[j+1])] = '>'
96         for c in FirstVT[VN.index(rule_list[0].left)]:
97             OG[VT.index('#')][VT.index(c)] = '<'
98         for c in LastVT[VN.index(rule_list[0].left)]:
99             OG[VT.index(c)][VT.index('#')] = '>'
100
101     def og():
102         with open('/home/aquawcac/code/OG/src.txt', 'r', encoding='utf-8') as src_file:
103             src = src_file.readlines()
104         for i in range(len(src)):
105             flag = False
106             og_stack = []
107             src[i] = src[i].replace('\n', '')
108             current = 0
109             pos = 1
110             og_stack.append('#')
111             while current != len(src[i]):
112                 a = src[i][current]
113                 s = og_stack[pos-1]
114                 print('%60s'%og_stack, '%20s'%a, '%20s'%src[i][current:], pos)
115                 if s in VT:
116                     j = pos
117                 else:
118                     j = pos - 1
119                 while pos != 2 or a != '#':
120                     if OG[VT.index(s)][VT.index(a)] == '>':
121                         while True:
122                             q = s
123                             j = j - 1
124                             s = og_stack[j-1]

```

```

125         if s not in VT:
126             j = j - 1
127             s = og_stack[j-1]
128         if OG[VT.index(s)][VT.index(q)] == '<':
129             pos = j + 1
130             if pos == len(og_stack):
131                 og_stack[pos-1] = 'N'
132             else:
133                 while pos - 1 != len(og_stack):
134                     og_stack.pop()
135                     og_stack.append('N')
136                 break
137         else:
138             og_stack.append(a)
139             pos = pos + 1
140             current = current + 1
141             break
142     if pos == 2:
143         if a == '#':
144             flag = True
145             break
146 with open('/home/aquawcac/code/OG/output.txt', 'a', encoding='utf-8') as out_file:
147     if flag:
148         out_file.write('%s 合法\n' % src[i])
149     else:
150         out_file.write('%s 不合法\n' % src[i])
151
152 def print_vt():
153     with open('/home/aquawcac/code/OG/set.txt', 'w', encoding='utf-8') as set_file:
154         set_file.write("FirstVT\n")
155         for k in range(len(VN)):
156             set_file.write("%3s:" % VN[k])
157             for p in FirstVT[k]:
158                 set_file.write("%s" % p)
159             set_file.write("\n")
160         set_file.write("LastVT\n")
161         for m in range(len(VN)):
162             set_file.write("%3s:" % VN[m])
163             for n in LastVT[m]:
164                 set_file.write("%s" % n)
165             set_file.write("\n")
166
167 def print_og():
168     with open('/home/aquawcac/code/OG/OG.txt', 'w', encoding='utf-8') as chart_write:
169         # chart_write.write('生成的优先矩阵如下\n')
170         chart_write.write("%3s" % '%')
171         for c in VT:
172             chart_write.write("%3s" % c)
173         chart_write.write("\n")
174         for i in range(len(OG)):
175             chart_write.write("%3s" % VT[i])
176             for j in range(len(OG[i])):
177                 chart_write.write("%3s" % OG[i][j])
178             chart_write.write("\n")
179
180 if __name__ == '__main__':
181     with open('/home/aquawcac/code/OG/output.txt', 'w', encoding='utf-8') as out_file:
182         pass
183     with open('/home/aquawcac/code/OG/rule.txt', 'r', encoding='utf-8') as rule_file:
184         Rules = rule_file.readlines()
185         for i in range(len(Rules)):
186             Rules[i] = Rules[i].replace('\n', '')
187     create_rule_list()
188     identify_vt_and_vn()
189     for j in range(len(VN)):
190         FirstVT.append([])
191         LastVT.append([])
192     OG = [[0 for col in range(len(VT))] for row in range(len(VT))]
193     for k in range(len(VN)):
194         gen_firstvt(VN[k])
195     for p in range(len(VN)):

```

```

196         gen_lastvt(VN[p])
197     print_vt()
198     create_og()
199     print_og()
200     og()

```

## 数据结构

### ST 表

- 二维

```

1  int f[maxn][maxn][10][10];
2  inline int highbit(int x) { return 31 - __builtin_clz(x); }
3  inline int calc(int x, int y, int xx, int yy, int p, int q) {
4      return max(
5          max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
6          max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
7      );
8  }
9  void init() {
10     FOR (x, 0, highbit(n) + 1)
11     FOR (y, 0, highbit(m) + 1)
12     FOR (i, 0, n - (1 << x) + 1)
13     FOR (j, 0, m - (1 << y) + 1) {
14         if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
15         f[i][j][x][y] = calc(
16             i, j,
17             i + (1 << x) - 1, j + (1 << y) - 1,
18             max(x - 1, 0), max(y - 1, 0)
19         );
20     }
21 }
22 inline int get_max(int x, int y, int xx, int yy) {
23     return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
24 }

```

## 数学

### 类欧几里得

- $m = \lfloor \frac{an+b}{c} \rfloor$ .
- $f(a, b, c, n) = \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor$ : 当  $a \geq c$  or  $b \geq c$  时,  $f(a, b, c, n) = (\frac{a}{c})n(n+1)/2 + (\frac{b}{c})(n+1) + f(a \bmod c, b \bmod c, c, n)$ ; 否则  $f(a, b, c, n) = nm - f(c, c-b-1, a, m-1)$ 。
- $g(a, b, c, n) = \sum_{i=0}^n i \lfloor \frac{ai+b}{c} \rfloor$ : 当  $a \geq c$  or  $b \geq c$  时,  $g(a, b, c, n) = (\frac{a}{c})n(n+1)(2n+1)/6 + (\frac{b}{c})n(n+1)/2 + g(a \bmod c, b \bmod c, c, n)$ ; 否则  $g(a, b, c, n) = \frac{1}{2}(n(n+1)m - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1))$ 。
- $h(a, b, c, n) = \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor^2$ : 当  $a \geq c$  or  $b \geq c$  时,  $h(a, b, c, n) = (\frac{a}{c})^2 n(n+1)(2n+1)/6 + (\frac{b}{c})^2 (n+1) + (\frac{a}{c})(\frac{b}{c})n(n+1) + h(a \bmod c, b \bmod c, c, n) + 2(\frac{a}{c})g(a \bmod c, b \bmod c, c, n) + 2(\frac{b}{c})f(a \bmod c, b \bmod c, c, n)$ ; 否则  $h(a, b, c, n) = nm(m+1) - 2g(c, c-b-1, a, m-1) - 2f(c, c-b-1, a, m-1) - f(a, b, c, n)$ 。

## 图论

### LCA

- 倍增

```

1  void dfs(int u, int fa) {
2      pa[u][0] = fa; dep[u] = dep[fa] + 1;
3      FOR (i, 1, SP) pa[u][i] = pa[pa[u][i-1]][i-1];
4      for (int& v: G[u]) {
5          if (v == fa) continue;
6          dfs(v, u);
7      }
8  }

```



```

9
10 int lca(int u, int v) {
11     if (dep[u] < dep[v]) swap(u, v);
12     int t = dep[u] - dep[v];
13     FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
14     FORD (i, SP - 1, -1) {
15         int uu = pa[u][i], vv = pa[v][i];
16         if (uu != vv) { u = uu; v = vv; }
17     }
18     return u == v ? u : pa[u][0];
19 }

```

## 计算几何

### 二维几何：点与向量

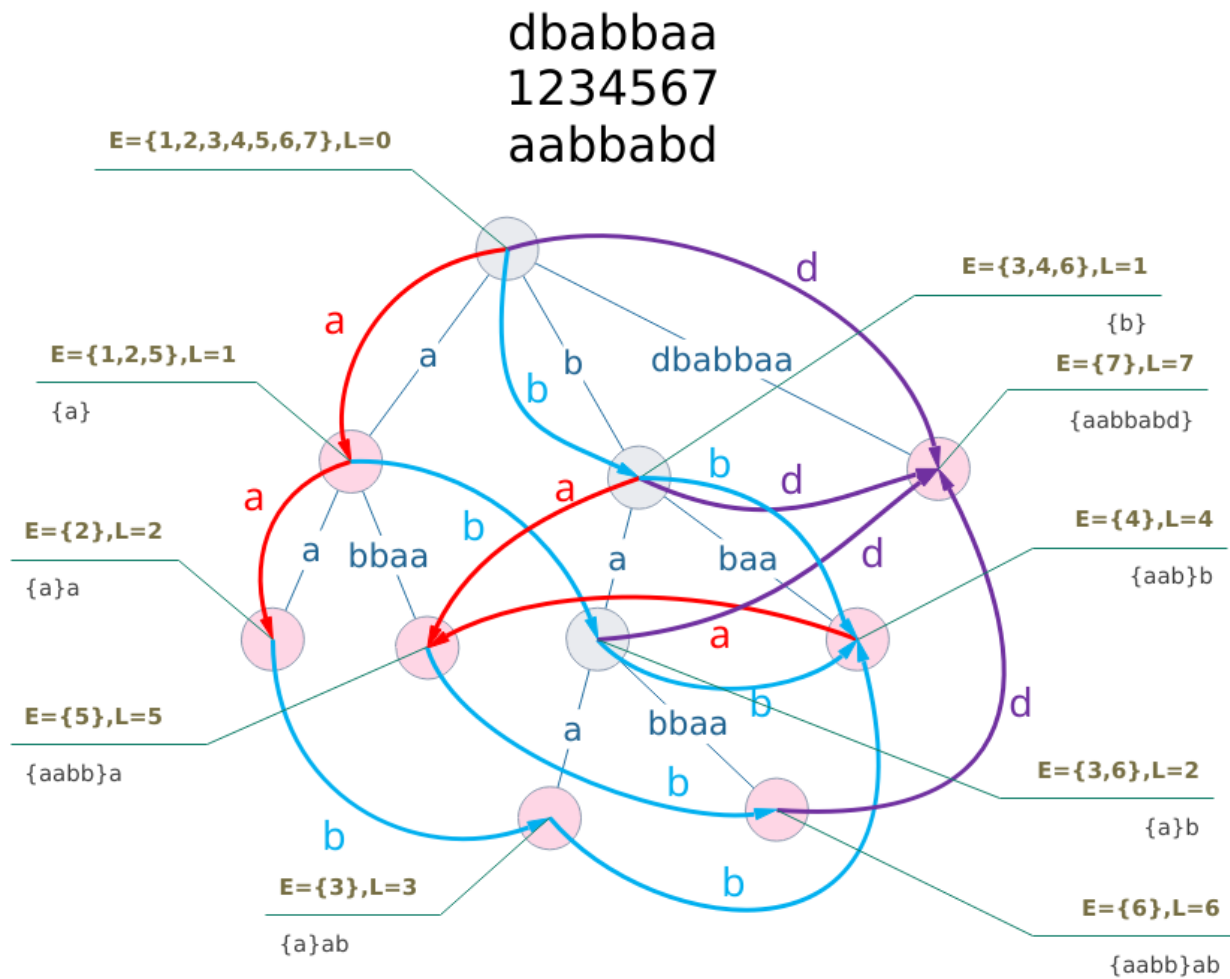
```

1  #define y1 yy1
2  #define nxt(i) ((i + 1) % s.size())
3  typedef double LD;
4  const LD PI = 3.14159265358979323846;
5  const LD eps = 1E-10;
6  int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
7  struct L;
8  struct P;
9  typedef P V;
10 struct P {
11     LD x, y;
12     explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
13     explicit P(const L& l);
14 };
15 struct L {
16     P s, t;
17     L() {}
18     L(P s, P t): s(s), t(t) {}
19 };
20
21 P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
22 P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
23 P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
24 P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
25 inline bool operator < (const P& a, const P& b) {
26     return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
27 }
28 bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
29 P::P(const L& l) { *this = l.t - l.s; }
30 ostream &operator << (ostream &os, const P &p) {
31     return (os << "(" << p.x << "," << p.y << ")");
32 }
33 istream &operator >> (istream &is, P &p) {
34     return (is >> p.x >> p.y);
35 }
36
37 LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
38 LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
39 LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
40 LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
41 // -----

```

## 字符串

### 后缀自动机



## 杂项

### STL

- copy

```
1 template <class InputIterator, class OutputIterator>  
2   OutputIterator copy (InputIterator first, InputIterator last, OutputIterator result);
```