YDSP 2024 · Senior 组初赛模拟考试

2024 年 9 月 14 日

本次考试的答题时间为两个小时，请各位选手自行掌握时间。

提交请访问 yundouxueyuan.com 参加对应的比赛，将最终答案以程序的形式提交至对应题目。

更进一步的提交细节，请参考对应题目里的说明。

最后，本次模拟考试为纯公益目的，不得用于任何未经授权的盈利活动。

* 祝考试顺利 *

2024 云斗学院软件能力认证第一轮

(YDSP - Senior) 提高级 C++ 语言试题

考生注意事项:

- 试题纸一共 14 页，满分 100 分。作答后请记录答案，并按要求提交至对应比赛处。
- 考试过程中不得使用任何电子设备或查阅任何书籍资料。

1 单选题（每题 2 分，共 30 分）

1.1 第 1 题

浙江省省队选拔的缩写为 ZJOI，其中 O 的中文含义是 __▲__。

A. O 神

B. 奥林匹克

C. 省队

D. 输出

1.2 第 2 题

方程组 $\begin{cases} 3a+2b+5c=3 \\ a+4b+c=17 \\ -a+6b+6c=4 \end{cases}$ 中，a 的值是 __▲__。

A. 3.2

B. 3

C. -4.2

D. 前三个选项都不对

1.3 第 3 题

关于树的直径和重心，下列说法正确的是 __▲__。

A. 一棵树最多有一条直径。

B. 树的直径必然穿过树的所有重心。

C. 偶数条边的树不可能有两个重心。

D. 若 $n \geq 2$ 且为自然数，则 $2^n - 1$ 个结点的树最多有 2^{2n-2} 条直径。

1.4 第 4 题

Alice 和 Bob 正在讨论排序算法。

- Alice：我在书上看到过，基于比较的排序算法时间复杂度不会低于 $\Theta(n \log n)$。
- Bob：可是 _____ 可以做到 $\Theta(n)$ 啊。
- Alice：我说的“时间复杂度”指的是平均情况，而不是最好情况。

根据上下文，Bob 最可能提到了 __▲__ 算法。

A. 插入排序

B. 快速排序

C. 基数排序

D. 堆排序

1.5 第 5 题

当输入的图是稀疏图，$m = \Theta(n)$ 时，可以使用一些数据结构来优化 Dijkstra 算法，让复杂度变成 $\Theta(n \log n)$。下面四个数据结构中，最合适的是 __▲__。

A. 单调队列

B. ST 表

C. 并查集

D. 线段树

1.6 第 6 题

STL 是我们写代码的好帮手，下列 STL 算法模板或容器拼写正确的是 __▲__。

A. `previous_permutation`

B. `priority_queue`

C. `kth_element`

D. `unordered_mutimap`

1.7 第 7 题

把 $2, 0, 2, 4, 0, 9, 2, 1$ 排成一个环，旋转后能重合的方案看成同一种，有 __▲__ 种排法。

A. 720

B. 5040

C. 3360

D. 420

1.8 第 8 题

使用两个栈 f, b 来实现一个队列，具体地，每次入队就将元素压进 b，出队就将元素从 f 弹出；特别地，若要出队时 f 为空，则将 b 栈中元素按照栈顶到栈底的顺序全部塞入 f 栈。设操作总次数为 n，那么，__▲__。

A. 出队操作每次时间复杂度为 $O(1)$。

B. 存在一种输入数据，使得总时间复杂度为 $O(n^2)$。

C. 单次入队或出队操作的均摊时间复杂度均为 $O(1)$。

D. 若还要实现 `pop_back`（即“若 b 为空则把 f 的元素塞入 b 栈，然后将 b 栈栈顶弹出”），不改变其它代码，仍然可以保证 n 次操作总时间复杂度 $O(n)$。

1.9 第 9 题

以边集数组的形式给出一张 n^k 个结点（假定 $2 \le k$ 且结点编号可以 $O(1)$ 参与运算），n 条边的图，希望任求一条欧拉回路，时间复杂度最低是 __▲__。

A. $O(n)$

B. $O(kn)$

C. $O(kn \log n)$

D. $O(n^k)$

1.10 第 10 题

一同学预处理了 `fact,invf` 数组，其中 `fact[i]` 表示 $i!$ 除以质数 $M = 10^9+7$ 的余数，`invf[i]` 表示 `fact[i]` 在模 $10^9 + 7$ 意义下的乘法逆元，那么下列说法正确的是 __▲__。

A. 要求出 `invf` 的前 n 项，时间复杂度最低为 $O(n \log n)$。

B. `invf[300]` 和 `300*invf[301]` 在模 $10^9 + 7$ 意义下同余。

C. 要计算从 2000 人中选出 500 人组合的方案数，可用 `1ll*fact[2000]*invf[1500]%M*invf[500]%M`。

D. 在 CSP 第二轮考试中，为了加速预处理过程，可以在程序内直接写出 $1!, 2!, \ldots, (10^6)!$ 的逆元。

1.11 第 11 题

一份文件仅含有英文小写字母，其中字母 $a, b, c, \ldots, z$ 出现次数依次为 $2^1, 2^2, \ldots, 2^{26}$，若使用二进制哈夫曼编码方式，则整份文件的哈夫曼编码总长度是 __▲__。

A. $5 \times 2^{26} - 10$

B. $2^{28} - 58$

C. $2^{28} - 56$

D. $2^{27} - 54$

1.12 第 12 题

宾果游戏是一款经典的游戏，玩法如下：

在 $n \times n$ 的棋盘上，每个格子都写有一个条件，玩家标记出所有自己满足条件的格子。如果某一条直线（一行、一列或一条对角线）上所有格子都被标记，则玩家获胜。

一玩家正在玩一款 $n = 6$ 的宾果游戏，并且失败了。他最多标记了 __▲__ 个格子。

A. 30

B. 29

C. 25

D. 前三个选项都不对

1.13 第 13 题

01 背包问题（有 n 个物品，每个物品有一个体积和价值。从中选出若干个，求体积不超过 V 前提下价值最大值）是 NP-Hard 的，这意味着 __▲__。

A. 人们证明了该问题不是 P 问题。

B. 可以在多项式时间内验证 01 背包问题一个解的正确性。

C. 该问题存在 DP 做法，因此该问题存在多项式时间复杂度解法。

D. 可以在多项式时间内，把哈密顿回路问题转化为 01 背包问题。

1.14 第 14 题

执行如下代码片段后，`*i` 的值为 __▲__。

```
set<int> s={50,10,20,30,20};
auto i= ++ ++ s.begin();
```

A. 10

B. 12

C. 20

D. 30

1.15 第 15 题

NOI Linux 2.0 中，拥有最高权限的用户是 __▲__。

A. `CCF_NOI`

B. `admin`

C. `root`

D. `Ka***5307`

2 阅读程序（无特殊说明时判断 1.5 分，选择 3 分。3 题共 40 分）

2.1 第 1 题（13 分）

阅读下面的程序，回答问题。

```
#include <bits/stdc++.h>

#define N 100010
#define ll long long
#define mod 998244353
#define end {puts("0");return 0;}

using namespace std;

inline ll rd(){
    char c;
    bool flag = false;
    while((c = getchar()) < '0' || c > '9')
        if(c == '-') flag = true;
    ll res = c - '0';
    while((c = getchar()) >= '0' && c <= '9')
        res = (res << 3) + (res << 1) + c - '0';
    return flag ? -res : res;
}
int n, b[N], c[N];

int main(){
    n = rd(); ll num = 0, ans = 1;
    for(int i = 1 ; i <= n ; i++) b[i] = rd();
    for(int i = 1 ; i <= n ; i++) c[i] = rd();
        int mx = c[1], mn = b[1];
    for(int i = 1 ; i <= n ; i++){
        if((i == 1 && b[i] != c[i]) ||
        (b[i] < b[i - 1] && c[i] > c[i - 1]))
        end
        if(i == 1) continue;
      if(b[i] > c[i]) end
        if(b[i] > mn) end
```

```
34          else if(c[i] < mx) end
35          else if(b[i] < mn){
36              num += mn - b[i] - 1, mn = b[i];
37              continue;
38          }
39          else if(c[i] > mx){
40              num += c[i] - mx - 1, mx = c[i];
41              continue;
42          }
43          if(!num) end
44          ans *= num ;
45      ans %= mod ;
46      num--;
47      }
48      printf("%lld\n", ans);
49      return 0;
50  }
```

输入数据满足 $1 \le n \le 10^5$，$1 \le b_i, c_i \le n$。试完成以下判断题和单选题：

2.1.1 判断题

1.（1 分）把第 6 行修改为 `#define end {puts("0");return;}`，程序的行为不变。

2. 删去第 43 行后，程序的行为不变。

3. 输入 `5\n5 4 3 2 1\n1 2 3 4 5` 时，输出为 `0`。

2.1.2 选择题

4. 当输入 `3\n1 1 1\n1 3 3` 时，输出为 __▲__ 。

A. 0

B. 1

C. 2

D. 3

5. 输入 `6\n3 3 1 1 1 1\n3 4 4 6 6 6` 时，若在第 37 行后要求输出 `num` 的值并删去第 40 行，则输出的结果为 __▲__。

A. 2，1

B. 1，2

C. 1，1

D. 2，2

6. 当 $n=10$，$b_1=c_1=c_2=5$，$b_i=1\ (i\geq 2)$，$c_i=10(i\geq 3)$ 时，输出为 ▲ 。

A. 720

B. 5040

C. 40320

D. 362880

2.2 第 2 题（13 分）

```
#include <bits/stdc++.h>

using namespace std;

#define int long long
#define endl '\n'

const int N = 1e3 + 5, INF = 1e18, M = 26;
int n, g[N], d[N]; string s[N];

int change(char c) {
  return c - 'a';
}
namespace A {
  int t[N][M], cnt, f[N], tot;
  void insert(string s, int i) {
    int p = 0;
    for(char c : s) {
      int k = change(c);
      if(!t[p][k])
        t[p][k] = ++cnt;
      p = t[p][k];
    }
    f[p] = i;
```

```
25      return;
26    }
27    void dfs(int u = 0) {
28      if(f[u])
29        g[f[u]] = ++tot, d[tot] = f[u];
30      for(char c = 'a'; c <= 'z'; c++)
31        if(t[u][change(c)]) dfs(t[u][change(c)]);
32      return;
33    }
34  };
35  namespace B {
36    vector<int> cnt[N];
37    void sort(vector<tuple<int, int>> &e) {
38      for(int k = 0; k < e.size(); k++) {
39        int v = get<0>(e[k]), i = get<1>(e[k]);
40        cnt[i].push_back(v);
41      }
42      e.clear();
43      for(int i = 1; i <= n; i++)
44        for(int v : cnt[i]) e.push_back({v, i});
45      for(int i = 1; i <= n; i++)
46        cnt[i].clear();
47      return;
48    }
49  };
50  namespace C {
51    int n = 26, m, x[N], y[N], b = 0, b1 = 0, b2 = 0, f = INF, h[N];
52    vector<tuple<int, int>> e[N];
53    void make(int u, int v, int i) {
54      u++, v++;
55      e[u].push_back({v, i});
56      x[u]++, y[v]++;
57      f = min(f, min(u, v));
58      return;
59    }
60    stack<int> t;
61    void dfs(int u) {
```

```
62      while(h[u] < e[u].size()) {
63        int q = d[get<1>(e[u][h[u]])];
64        dfs(get<0>(e[u][h[u]++])), t.push(q);
65      }
66      return;
67    }
68    void work() {
69      for(int i = 1; i <= n; i++)
70        if(x[i] == y[i]) b++;
71        else if(x[i] - y[i] == 1) b1++, f = i;
72        else if(y[i] - x[i] == 1) b2++;
73        else cout<<"No Solution!"<<endl, exit(0);
74      if(!(b == n || (b1 == 1 && b2 == 1)))
75        cout<<"No Solution!"<<endl, exit(0);
76      for(int i = 1; i <= n; i++) B::sort(e[i]);
77      dfs(f); assert(t.size() == m);
78      while(!t.empty()) cout<<s[t.top()], t.pop(); cout<<endl;
79      return;
80    }
81  };
82  signed main() {
83    cin>>n; C::m = n;
84    for(int i = 1; i <= n; i++) cin>>s[i];
85    for(int i = 1; i <= n; i++) A::insert(s[i], i);
86    A::dfs();
87    for(int i = 1; i <= n; i++)
88      C::make(change(s[i].front()), change(s[i].back()), g[i]);
89    C::work();
90    return 0;
91  }
```

令 $\sum |s_i| = m$。输入数据满足 $1 \le n, m \le 10^3$，且字符串仅含有小写字母。

2.2.1 判断题

1.（1 分）将第 5 行删去，程序的行为不变。

2.（1 分）将第 12 行的 `c - 'a'` 改为 `c - 'a' + 1`，程序的行为不变。

3. 将第 39 行的 `get<0>(e[k])` 改为 `e[k].first`，程序的行为不变。

4. 程序总是能正常运行。

2.2.2 选择题

5. （2 分）当输入为 `3\naab\naza\naea` 时，输出为 __▲__。

   A. `azaaeaaab`

   B. `aabaeaaza`

   C. `aaaaaabez`

   D. `aeaazaaab`

6. 执行第 86 行后，n 和 `namespace A` 中的 tot 的大小关系为 __▲__。

   A. 总是 $n = \text{tot}$

   B. 有时 $n < \text{tot}$，有时 $n = \text{tot}$

   C. 有时 $n > \text{tot}$，有时 $n = \text{tot}$

   D. 无法确定

7. 我们认为 n, m 同阶，字符集大小为 k，则程序的时间复杂度为 __▲__。

   A. $\mathcal{O}(n)$

   B. $\mathcal{O}(n^2)$

   C. $\mathcal{O}(k \cdot n)$

   D. $\mathcal{O}(k \cdot n^2)$

2.3 第 3 题（14 分）

```
#include<bits/stdc++.h>
using namespace std;
#define Ull unsigned long long
const int Mul=29,L=10000005;
char a[L];
Ull pre[L],suf[L],pw[L];
Ull prehash(int l,int r){
  l--;
  Ull res=pre[l]*pw[r-l];
  return pre[r]-res;
```

```
}
Ull sufhash(int l,int r){
  r++;
  Ull res=suf[r]*pw[r-l];
  return suf[l]-res;
}
int main(){
  scanf("%s",a+1);
  int n=strlen(a+1);
  pw[0]=1;
  for(int i=1;i<=n;i++)
    pw[i]=pw[i-1]*Mul;
  for(int i=1;i<=n;i++)
    pre[i]=pre[i-1]*Mul+a[i];
  for(int i=n;i;i--)
    suf[i]=suf[i+1]*Mul+a[i];
  int l=1,r=0,res1=0;
  while(r<=n){
    while(l && prehash(l,r)==sufhash(l,r))
      l--,r++;
    l++,r--;
    res1=r-l+1;
    l++,r++;
    while(r<=n && prehash(l,r)!=sufhash(l,r))
      l++,r++;
  }
  l=1,r=1;
  int res2=1;
  while(r<=n){
    while(l && prehash(l,r)==sufhash(l,r))
      l--,r++;
    l++,r--;
    res2=r-l+1;
    l++,r++;
    while(r<=n && prehash(l,r)!=sufhash(l,r))
      l++,r++;
  }
```

```
48      printf("%d",max(res1,res2));
49      return 0;
50  }
```

若无特殊说明，保证输入的字符串包含且仅包含小写字母，且长度在 $[1,10^7]$ 之间。

2.3.1 判断题

1. 若去掉第 33 行，程序可能死循环。
2. 若随机生成一个长度为 10^5 字符的字符串作为输入，则在第 27 运行结束后，等式 `prehash(921,2024)==sufhash(921,2024)` 成立的概率约为 10^{-7}。
3. （2 分）程序时间复杂度为 $O(n)$，其中 n 为字符串长度。

2.3.2 选择题

4. （2 分）假设输入是长 10^7 的、每个字符都是 `a` 或 `b` 中均匀选取的字符串。进行下面 __▲__ 选项的改动后，代码的输出发生改变的概率最大。

   A. 把第 3 行改成 `#define Ull unsigned`

   B. 把 `Mul` 改成 64

   C. 把第 38 行改成 `int res2=3;`

   D. 把第 19 行改成 `int n=strlen(a+1)+2;`

5. 输入为 `yummymmyummyisatyundou` 时，输出为 __▲__。

   A. 5

   B. 4

   C. 3

   D. 2

6. （4 分）有 __▲__ 个含 5 个小写字母的输入，使得输出为 4。

   A. 35126

   B. 35152

   C. 17576

   D. 前三个选项都不对

3 完善程序（2 题，每空 3 分，共 30 分）

3.1 非空好子串统计

给定数列 $\{a_n\}$ 。数列 $\{b_m\}$ 是好的，当且仅当它能被划分成若干个子序列，满足每个子序列都构成一个有序的排列。求数列 $\{a_n\}$ 有多少个好的子串。

已知：$1 \le n \le 10^5$，$1 \le a_i \le n$。

提示：考虑转化原问题。倒序加入数字。加入一个数字时，考虑合并两个数。使用栈来维护。

试补全以下程序。

```
#include <iostream>
#include <stack>

using namespace std;

const int N = 100005;
int n, a[N];
int del[N];
long long ans;
stack<int> res, p[N];

int main() {
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> a[i];

    res.push(/* Blank 1 */);

    for (int i = n; i >= 1; --i) {
        if (/* Blank 2 */) {
            ++del[p[a[i] + 1].top()];
            p[a[i] + 1].pop();
        }
        if (/* Blank 3 */) {
            res.push(i);
            /* Blank 4 */;
        }
        while (del[res.top()]) res.pop();
        ans += /* Blank 5 */;
```

```
29      }
30
31      cout << ans << endl;
32      return 0;
33  }
```

1. 第 /* Blank 1 */ 空应该填 __▲__。

   A. `n`

   B. `1`

   C. `n + 1`

   D. `a[n]`

2. 第 /* Blank 2 */ 空应该填 __▲__。

   A. `!p[a[i] + 1].empty()`

   B. `p[a[i] + 1].empty()`

   C. `!p[a[i]].empty()`

   D. `!p[a[i] - 1].empty()`

3. 第 /* Blank 3 */ 空应该填 __▲__。

   A. `a[i] > 1`

   B. `i != n`

   C. `i != 1`

   D. `!p[a[i]].empty()`

4. 第 /* Blank 4 */ 空应该填 __▲__。

   A. `-- del[a[i]];`

   B. `p[a[i]].push(i);`

   C. `p[a[i]].pop();`

   D. `-- del[a[i] + 1];`

5. 第 /* Blank 5 */ 空应该填 __▲__。

   A. `res.size()`

   B. `res.top()`

   C. `max(0 , res.size() - i + 1)`

   D. `res.top() - i`

3.2 网格图上路径转合法括号序列

给定长度为 $2n$ 的仅包含 R（向右）和 D（向下）的字符串，表示一条 $(0,0)$ 到 (n,n) 的路径。可以证明，合法路径一共有 $\binom{2n}{n}$ 种，长度为 $2n$ 的合法括号序列一共有 $\frac{\binom{2n}{n}}{n+1}$ 种。

上述信息意味着，存在构造函数 f，使得对于每一种合法路径，都可以构造出不同的元素对 (x,y)。其中 x 为一个长度为 $2n$ 的合法括号序列，y 是一个不超过 n 的自然数，以下程序实现了一种构造函数 f 。

括号序列是一个仅由 `()` 构成的序列。以下的括号序列是合法的：

1. `()` 是一个合法括号序列。
2. 如果 `A` 是一个合法括号序列，则 `(A)` 也是一个合法括号序列。
3. 如果 `A` 和 `B` 都是合法括号序列，则 `AB` 也是一个合法括号序列。

已知： $1 \le n \le 100$ ， $s_i \in \{\mathrm{R},\mathrm{D}\}$ 。

提示：可以通过计算字典序来形成对应关系。对于括号序列的计算字典序的问题，可以考虑把括号序列转化，例如 `(()(()))` 可以转化为 $\{3,2,2,1\}$ ，然后用 $f_{i,j}$ 表示转化后的序列长为 i 结尾元素为 j 的序列有多少，用 $s_{i,j}$ 表示 $f_{i,j}$ 的前缀和，同时利用组合数。

试补全程序。

```
#include<iostream>
#include<cstring>
using namespace std;

const int N = 303, base = 1e4;

struct BigNum {
  int len, a[N];

  BigNum(const int& x = 0) {
    memset(a, 0, sizeof a);
    len = 0;
    if (x > 0) {
      len = 1;
      a[0] = x;
    }
  }

  BigNum operator + (const BigNum& b) {
    BigNum c;
    c.len = max(len, b.len) + 1;
```

```
22      int x, y = 0;
23      for (int i = 0; i < c.len; ++i) {
24        x = a[i] + b.a[i] + y;
25        c.a[i] = x % base;
26        y = x / base;
27      }
28      while (c.len && c.a[c.len - 1] == 0) --c.len;
29      return c;
30    }
31
32    BigNum operator - (const BigNum& b) {
33      BigNum c;
34      c.len = max(len, b.len);
35      int x, y = 0;
36      for (int i = 0; i < c.len; ++i) {
37        x = a[i] - b.a[i] + y;
38        if (x < 0) /* Blank 1 */;
39        else c.a[i] = x, y = 0;
40      }
41      while (c.len && c.a[c.len - 1] == 0) --c.len;
42      return c;
43    }
44
45    BigNum operator * (const int& b) {
46      BigNum c;
47      c.len = len + 1;
48      int x, y = 0;
49      for (int i = 0; i < c.len; ++i) {
50        x = a[i] * b + y;
51        c.a[i] = x % base;
52        y = x / base;
53      }
54      while (c.len && c.a[c.len - 1] == 0) --c.len;
55      return c;
56    }
57
58    pair<BigNum, int> div(const int& b) {
```

```
59      BigNum c;
60      c.len = len;
61      int x, y = 0;
62      for (int i = c.len - 1; i >= 0; --i) {
63        x = y * base + a[i];
64        c.a[i] = x / b;
65        y = x % b;
66      }
67      while (c.len && c.a[c.len - 1] == 0) --c.len;
68      return make_pair(c, y);
69    }
70
71    bool operator <= (const BigNum& b) {
72      if (len < b.len) return 1;
73      if (len > b.len) return 0;
74      for (int i = len - 1; i >= 0; --i) {
75        if (a[i] < b.a[i]) return 1;
76        if (a[i] > b.a[i]) return 0;
77      }
78      return 1;
79    }
80  };
81
82  BigNum c[N * 2][N], f[N][N], s[N][N];
83  int n, k;
84  string a;
85  int p[N];
86
87  int main() {
88    cin >> n >> a;
89
90    for (int i = 1; i <= n; ++i) {
91      f[1][i] = 1;
92      s[1][i] = s[1][i-1] + 1;
93    }
94
95    for (int i = 2; i <= n; ++i) {
```

```
96        for (int j = 1; j <= n; ++j) {
97          f[i][j] = /* Blank 2 */;
98          s[i][j] = s[i][j - 1] + f[i][j];
99        }
100     }
101
102     c[0][0] = 1;
103     for (int i = 1; i <= n << 1; ++i) {
104       c[i][0] = 1;
105       for (int j = 1; j <= i && j <= n+1; ++j)
106         c[i][j] = /* Blank 3 */;
107     }
108
109     BigNum num = 0;
110     int cnt = 0;
111
112     for (int i = 0; i < n << 1; ++i) {
113       if (a[i] == 'R') {
114         ++cnt;
115         num = num + /* Blank 4 */;
116       }
117     }
118
119     pair<BigNum, int> res = num.div(n + 1);
120     num = res.first;
121     int r = res.second;
122
123     for (int i = 1; i <= n; ++i) {
124       p[i] = 1;
125       for (int j = 1; j <= p[i-1]; ++j) {
126         BigNum value = /* Blank 5 */;
127         if (value <= num) {
128           num = num - value;
129           ++p[i];
130         }
131       }
132     }
```

```
133
134     for (int i = 1; i <= n; ++i) {
135       for (int j = p[i]; j <= p[i-1]; ++j)
136         cout << ')';
137       cout << '(';
138     }
139
140     for (int i = 1; i <= p[n]; ++i) cout << ')';
141     cout << '\n' << r << '\n';
142
143     return 0;
144   }
```

1. 第 /* Blank 1 */ 空应该填 __▲__。

   A. `c.a[i] = x + base, y = -1`

   B. `c.a[i] = x - base, y = 1`

   C. `c.a[i] = x + base * y, y = -y`

   D. `c.a[i] = x - base * y, y = -y`

2. 第 /* Blank 2 */ 空应该填 __▲__。

   A. `s[i - 1][j - 1]`

   B. `s[i - 1][j + 1]`

   C. `s[i + 1][j - 1]`

   D. `s[i + 1][j]`

3. 第 /* Blank 3 */ 空应该填 __▲__ 。

   A. `c[i][j - 1] + c[i - 1][j - 1]`

   B. `c[i - 1][j - 1] + c[i - 1][j] + c[i][j - 1]`

   C. `c[i - 1][j] + c[i][j - 1]`

   D. `c[i - 1][j] + c[i - 1][j - 1]`

4. 第 /* Blank 4 */ 空应该填 __▲__。

   A. `c[2 * n - i + 1][n - cnt -1]`

   B. `c[2 * n - i - 1][n - cnt + 1]`

C. `c[2 * n - i][n - cnt]`

D. `c[2 * n - i + 1][n - cnt - 1]`

5. 第 /* Blank 5 */ 空应该填 __▲__。

A. `f[n - i + 1][j]`

B. `f[n - i - 1][j - 1]`

C. `f[n - i + 1][j - 1]`

D. `f[n - i - 1][j + 1]`