一、选择题（每题2分，共计30分）

1、在一座办公室楼内各室计算机连成网络属于（）。 B

A、WAN

B、LAN

C、MAN

D、GAN

2、汇编语言是一种（）程序设计语言。 C

A、与具体计算机无关的高级 B、面向对象的

C、依赖于具体计算机的低级 D、面向过程的

3、下列说法错误的是（C）。

A、栈和队列的存储方式既可是顺序方式，也可是链式方式

B、线性表在物理存储空间中不一定是连续的。

C、二叉树中每个节点的两棵子树高度差为1

D、对于一棵非空二叉树，它的根节点作为第一层，则第i层至多有2^(i-1)个节点

4、输入序列为ABC，可以变为CBA时，经过的栈操作为（）。B

A、push,pop,push,pop,push,pop

B、push,push,push,pop,pop,pop

C、push,push,pop,pop,push,pop

D、push,pop,push,push,pop,pop

5、将 24 个志愿者名额分配给 3 个学校，则每校至少有一个名额且各校名额互不相同的分配方法共有（）种。 C

A.145 B.183 C.222 D.253

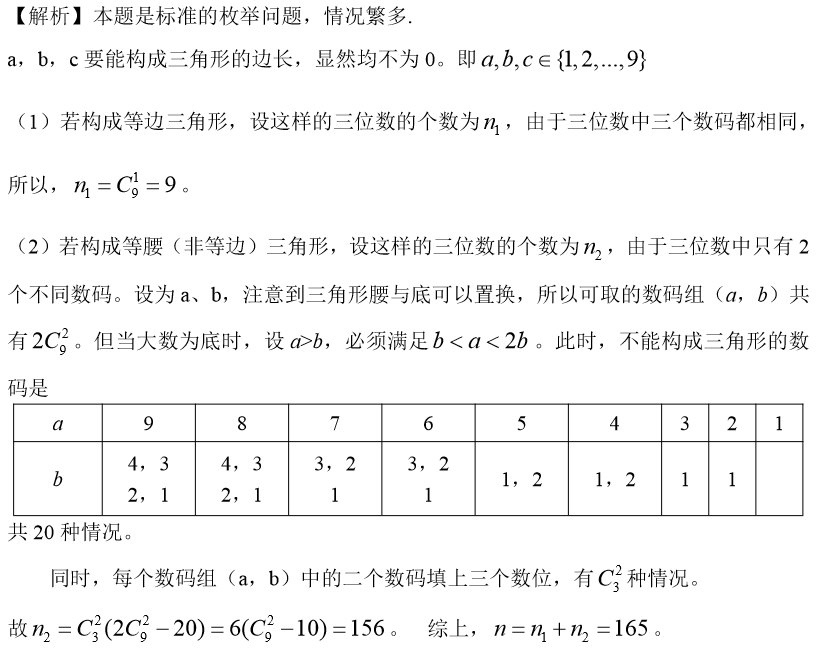
挡板原理：设分配给 3 个学校的名额数分别为 x1 x2 x3 ，则每校至少有一个名额的分法数为不定方程 x1+x2+x3=21的非负整数解的个数，它等于 3 个不同元素中取 21 个元素的可重组合：

C(21,23)=C(2,23)=253 ．又在“每校至少有一个名额的分法”中“至少有两个学校的名额数相同”的分配方法有 31 种．综上知，满足条件的分配方法共有 253－31＝222 种

6、设三位数abc ，若以 a，b，c 为三条边的长可以构成一个等腰（含等边）三角形，则这样的三位数 n 有（ ）C

A.45 个 B.81 个 C.165 个 D.216 个

解析：



7、十进制算术表达式：3\*512+7\*64＋4\*8＋5的运算结果，用二进制表示为（ ）． B

A. 10111100101 B.11111100101

C. 11110100101 D.11111101101

8、下列说法正确的是（）。B

A．使用SPFA判断n个节点图是否存在负环，只要松弛次数大于等于n，就一定存在

B．最小生成树保证生成树中u到v路径上的边权最大值一定**不大于**其他从u到v路径的边权最大值

C．若图有重边，将无法使用Kruskal算法求求最小生成树。

D．若图有负权边，一定无法用dijkstra算法求最短路。

A不一定，可能存在重边，应该用入队次数判更保险

9、一个8位二进制整数采用补码表示，且由3个“1”与5个“0”组成，则最小值为（）。C

1. -127 B、-32 C、-125 D、-3

10、计算机操作的最小时间单位是（）。A

A、时钟周期 B、指令周期 C、CPU周期 D、外围设备周期

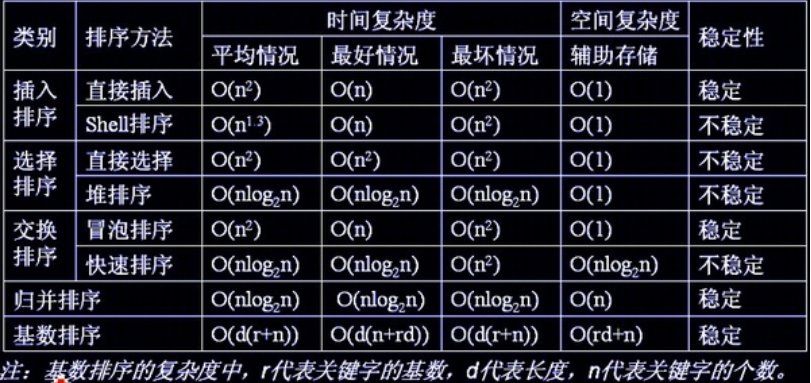
11、下列排序算法中最好最坏情况下的时间复杂度相同的是（）。C

A、希尔排序

B、快速排序

C、堆排序

D、基数排序



12、下列关于说法正确的是（）。 B

A、贪心算法不具有最优子结构性质

B、动态规划全局最优解一定包含局部最优解

C、动态规划可以不满足无后效性

D、贪心算法的子问题往往互相独立

13、已知某程序的时间复杂度的递推公式为：T(n)=25T(n/5)+n^2，求T(n)？ B

A、O(n^2) B、 O(n^2 logn)

C、O(nlog^2 n) D、O(n^3)

14、本题中，我们约定布尔表达式只能包含p,q两个布尔变量，以及“与”（∧）、“或”（∨）、“非”（¬）三种布尔运算。如果无论p, q如何取值，两个布尔表达式的值总是相同，则称它们等价。例如，p∨¬p 和q∨¬q 等价； 而p∨q 和p∧q不等价。那么，两两不等价的布尔表达式最多有（ ）个。 D

A、2 B、4

C、8 D、16

两个变量，每个变量可取0,1两种值，共有2^2=4种组合；任意一个变量组合代入表达式，只有0和1两种值。因此两两不等价的表达式最多有2^4=16种。

15、数列连续三项an+2，an+1，an满足：2an+2 -5an+1 +3an=0 (n>=1)，a1=2，a2=1，那么当i增大时，ai将趋向于（） A

A、负无穷

B、负根号5

C、-2

D、0

解析：二阶特征方程可解。

三、阅读程序写结果（共3题）

1、

1 #include<iostream>

2 using namespace std;

3 void fun(char \*a, char \*b, int &c) {

4 a = b;

5 (\*a)++;

6 c++;

7 }

8 int main() {

9 char c1, c2, \*p1, \*p2;

10 c1 = 'A';

11 c2 = 'a';

12 int c3;

13 c3 = 48;

14 p1 = &c1;

15 p2 = &c2;

16 fun(p1, p2, c3);

17 cout << c1 <<" "<< c2 <<" "<< c3 << endl;

18 return 0;

19 }

16、第3行和第14行的‘&’含义相同。（ ） 否

17、去掉第三行的‘&’，输出结果不变。（ ） 否

18、去掉第5行的( )，输出结果不变。 （ ）否

19、第3行和第6行的c可以改成c3且不影响结果。 （ ）是

20、程序输出结果为：（ ）。 D

A．b a 49

B．B b 48

C．A a 48

D．A b 49

21、第4行改成“\*a=\*b”，程序输出结果为：（ ）。 B

A．B a 49

B．b a 49

C．A b 48

D．A a 49

2、

#include<bits/stdc++.h>

using namespace std;

int a[40][40];

int main() {

int n,x[1600],y[1600];

scanf("%d",&n);

for(int i=0; i<=n; i++) {

a[0][i]= 2147483647,a[i][0]= 2147483647;

}

a[1][n/2+1]=1;

x[1]=1,y[1]=n/2+1;

for(int i=2; i<=n\*n; i++) {

if(x[i-1]==1&&y[i-1]!=n) {

a[n][y[i-1]+1]=i;

x[i]=n,y[i]=y[i-1]+1;

} else if(y[i-1]==n&&x[i-1]!=1) {

a[x[i-1]-1][1]=i;

x[i]=x[i-1]-1,y[i]=1;

} else if(a[1][n]==i-1) {

a[2][n]=i;

x[i]=2,y[i]=n;

} else if(x[i-1]!=1&&y[i-1]!=n) {

if(a[x[i-1]-1][y[i-1]+1]==0) {

a[x[i-1]-1][y[i-1]+1]=i;

x[i]=x[i-1]-1,y[i]=y[i-1]+1;

} else {

a[x[i-1]+1][y[i-1]]=i;

x[i]=x[i-1]+1,y[i]=y[i-1];

}

}

}

for(int j=1; j<=n; j++)

printf("%d ",a[3][j]);

return 0;

}

22、2147483647是int范围内最大的合数。（ ） 否

23、第7行i=0改为i=1，不影响输出结果。（ ） 是

24、对于任意输入，a[1][1]一定为1。 （ ）否

25、对于任意输入，程序输出的序列数字一定不同。 （ ）是

26、n为3，程序输出结果为：（ ）。 D

A．5 9 3

B．4 6 2

C．5 7 3

D．4 9 2

27、n为5，程序输出结果为：（ ）。 B

A．5 6 7 1 4

B．4 6 13 20 22

C．3 6 9 18 12

D．4 10 6 10 32

3、

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

2 using namespace std;

3 vector<char> v;

4 stack<char> path;

5 list<char> adj[1000];

6 void trav(int start) {

7 path.push(start);

8 while(path.size()) {

9 char top = path.top();

10 if(adj[top].size()) {

11 path.push(adj[top].back());

12 adj[adj[top].back()].remove(top);

13 adj[top].pop\_back();

14 } else {

15 v.push\_back(top);

16 path.pop();

17 }

18 }

19 }

20 int main() {

21 int n;

22 cin >> n;

23 while(n--) {

24 char c1, c2;

25 cin >> c1 >> c2;

26 adj[c1].push\_back(c2);

27 adj[c2].push\_back(c1);

28 }

29 int even = 0, odd = 0, start = 'a';

30 for(int i = 'z'; i >= 'A'; i--) {

31 if(adj[i].size()%2) {

32 odd++;

33 start=i;

34 } else

35 even++;

36 }

37 if(odd==0||odd==2) {

38 trav(start);

39 for(int i = v.size()-1; i >= 0; i--)

40 cout << char(v[i]);

41 cout << endl;

42 } else

43 cout << "No Solution" << endl;

44 }

28、程序第4行stack改为queue不会影响程序结果。 （ ）否

29、29行start值改为1，不影响输出结果。（ ） 否

30、even为偶数时，程序一定不会输出“No Solution”。 （ ）否

31、n=2时，对于任意输入，程序一定不会输出“No Solution”。 （ ）否

32、若输入为：

4

aZ

tx

Xz

aX

程序输出结果为：（ ）。 D

A．aaZzX

B．aZzXa

C．aXaZz

D．No Solution

33、若输入为：

4

aZ

tZ

Xt

aX

程序输出结果为：（ ）。 D

A．XXZat

B．taZXX

C．XZatX

D．XaZtX

XJOI 8264

三、完善程序（共2题，10空，共40分）

1、给一个长度为n的排列p，找一个元素，使得从排列中取出这个元素以后排列的“records”最多。

一个"record"是一个元素ai满足：对于每个正整数1≤j<i , ai > aj​

(1≤n≤1e5)

Codeforces 900C. Remove Extra One

#include<bits/stdc++.h>

using namespace std;

int n,a[100010],cnt[100010];

int main() {

int max1=0,max2=0; //记录最大值、次大值

scanf("%d",&n);

for(int i=1; i<=n; i++)

scanf("%d",&a[i]);

for(int i=1; i<=n; i++)

if( 1 ) {

2 ;

max1=a[i];

3 ;

} else if( 4 ) {

cnt[max1]++;

5 ;

}

int ans,max3=-100000;

for(int i=1; i<=n; i++)

if(cnt[i]>max3) {

max3=cnt[i];

ans=i;

}

printf("%d\n",ans);

}

因为最大值可以删去，所以我们关心的不仅只有最大值，还有次大的

一组数字，没有修改的话原本的贡献是相同的，所以问题就变成了去掉一个数最多能增加多少贡献

这该怎么记录呢？

如果有一个数比当前最大数大，那么去掉它会产生负贡献

如果比最大值小，比次大值大，那么去掉它最大值会增加一个贡献

所以建一个cnt数组，cnt[i]表示去掉其所增加的贡献

34、1处填（ ）。A

A．a[i]>max1 B．a[i]<max1

C. a[i]>max2 D．a[i]<max2

35、2处填（ ）。B

A．max1=max2 B．max2=max1

C. max2=a[i] D．max1--

36、3处填（ ）。C

A．cnt[a[i]]++ B． cnt[max2]++

C. cnt[a[i]]-- D．cnt[max1]--

37、4处填（ ）。C

A．a[i]>max1 B．a[i]<max1

C. a[i]>max2 D．a[i]<max2

38、5处填（ ）。C

A．max1=max2 B．max2=max1

C. max2=a[i] D．max1--

2、

有一颗 n 个节点的树，k 次旅行，问每一条边被走过的次数。

输入格式：

第一行一个整数n（2≤n≤10^5）。

接下来 n-1行，每行两个正整数 x,y （1≤x,y≤n,x!=y），表示x与y之间有一条连边。

接下来一个整数 k （0≤k≤10^5）。

接下来 k 行，每行两个正整数x,y （1≤x,y≤n ），表示有一个从x到y的旅行。

输出格式：

总共 n-1行，按输入顺序输出每一条边被走过的次数。

CF191C Fools and Roads

#include<bits/stdc++.h>

using namespace std;

struct Edge {

int to;

int nxt;

int head;

};

Edge edge[200005];

int n,m,u,v,lg[100005],x[100005],y[100005],point,anc[100005][20],depth[100005],diff[100005];

inline void init\_log() {

lg[1]=1;

for(register int i=2; i<=n; i++) {

if(1<<lg[i-1]==i)

lg[i]=1;

lg[i]+=lg[i-1];

}

}

inline void add\_edge(int u,int v) {

point++;

edge[point].to=v;

edge[point].nxt=edge[u].head;

edge[u].head=point;

}

inline void dfs\_anc(int now,int father) {

anc[now][0]=father;

depth[now]=depth[father]+1;

for(register int i=1; i<=lg[depth[now]]; i++) {

anc[now][i]= 1 ;

}

int tmp=edge[now].head;

while(tmp>0) {

int son=edge[tmp].to;

if(son==father) {

tmp=edge[tmp].nxt;

continue;

}

dfs\_anc(son,now);

tmp=edge[tmp].nxt;

}

}

inline int lca(int a,int b) {

if(depth[a]<depth[b])

swap(a,b);

while(depth[a]>depth[b])

a= 2 ;

if(a==b) return b;

for(register int i= 3 ; i>=0; i--)

if(anc[a][i]!=anc[b][i]) {

a=anc[a][i];

b=anc[b][i];

}

Return 4 ;

}

inline void dfs\_answer(int now,int father) {

int tmp=edge[now].head;

while(tmp>0) {

int son=edge[tmp].to;

if(son==father) {

tmp=edge[tmp].nxt;

continue;

}

dfs\_answer(son,now);

diff[now]+=diff[son];

tmp=edge[tmp].nxt;

}

}

int main() {

scanf("%d",&n);

init\_log();

for(register int i=1; i<n; i++) {

scanf("%d%d",&x[i],&y[i]);

add\_edge(x[i],y[i]);

add\_edge(y[i],x[i]);

}

dfs\_anc(1,0);

scanf("%d",&m);

for(register int i=1; i<=m; i++) {

scanf("%d%d",&u,&v);

diff[u]++;

diff[v]++;

5 ;

}

dfs\_answer(1,0);

for(register int i=1; i<n; i++) {

u=x[i],v=y[i];

if(depth[u]>depth[v])

printf("%d ",diff[u]);

else

printf("%d ",diff[v]);

}

return 0;

}39、1处填（ ）。B

A．0 B．anc[anc[now][i-1]][i-1]

C. 1 D．anc[now][i]+anc[now][i-1]

40、2处填（ ）。D

A．lg[depth[a]-depth[b]]-1 B．depth[a]-depth[b]-1

C. depth[a]-depth[b]-1 D．anc[a][lg[depth[a]-depth[b]]-1]

41、3处填（ ）。D

A．depth[a] B．depth[a]-1

C. lg[depth[a]] D．lg[depth[a]]-1

42、4处填（ ）。C

A．1 B．anc[depth[a]][0]

C. anc[a][0] D．anc[depth[a]][ depth[b]]

43、5处填（ ）。C

A. diff[lca(u,v)]-=1 B．diff[lca(u+1,v-1)]-=1

C. diff[lca(u,v)]-=2 D．diff[lca(u+1,v-1)]-=2