**好的开始**

1. #include<bits/stdc++.h>
2. **using** **namespace** std;
3. #define \_for(i,a,b) for(int i=(a);i<(b);i++)
4. #define \_ref(i,a,b) for(int i=(a);i<=(b);i++)
5. #define ms(a, b) memset(a, b, sizeof(a))
6. #define fill(a,v,n) memset((a),(v),sizeof(a[0])\*(n))
7. #define copy(a,b,n) memcpy((a),(b),sizeof(a[0])\*(n))
8. #define sc(x) scanf("%d", &x)
9. #define pr(x) printf("%d", (x))
10. #define scl(x) scanf("%lld", &x)
11. #define prl(x) printf("%lld", (x))
12. #define sp putchar(' ')
13. #define en putchar('\n')
14. #define ALL(x) (x.begin(),x.end())
15. #define pb push\_back
16. #define mp make\_pair
17. #define fi first
18. #define se second
19. #define IOS ios::sync\_with\_stdio(0);cin.tie(0);
20. **typedef** **double** db;
21. **typedef** **long** **long** ll;
22. **typedef** unsigned **long** **long** ull;
23. **typedef** pair<**int**, **int**> pii;
24. **typedef** pair<ll, ll> pll;
26. #define MOD 1000000007
27. **void** \_max(**int** &a, **int** b){a = max(a,b);}
28. **void** \_min(**int** &a, **int** b){a = min(a,b);}
30. ll pw(ll a, ll b){ll res(1);**while**(b){**if**(b&1)res=res\*a%MOD;a=a\*a%MOD;b>>=1;}**return** res;}
31. ll gcd(ll a, ll b){ll t;**while**(b){t=a%b;a=b;b=t;}**return** a;}
32. ll lcm(ll a, ll b){**return** a/gcd(a,b)\*b;}
33. **int** len(ll x){**int** k=0;**while**(x){x/=10;k++;}**return** k;}
34. **int** len(**int** x){**int** k=0;**while**(x){x/=10;k++;}**return** k;}
36. **const** **int** infi = 2147483647;
37. **const** ll infl = 9223372036854775807ll;
38. **const** db PI = 3.14159265358979323846;
40. **inline** **void** rd(**int** &x){x = 0;**char** c = getchar();**while**(c<'0'||c>'9')c=getchar();**while**(c>='0'&&c<='9')x=x\*10+(c&15),c=getchar();}
41. **inline** **void** llread(ll &x){**char** ch = getchar(); x = 0;**for** (; ch < '0' || ch > '9'; ch = getchar());**for** (; ch >= '0' && ch <= '9'; ch = getchar()) x = x \* 10 + ch - '0';}
43. #define LOCAL\_JUDGE
45. **int** main()
46. {
47. #ifdef LOCAL\_JUDGE
48. freopen("Text.txt", "r", stdin);
49. #endif // LOCAL\_JUDGE
51. #ifdef LOCAL\_JUDGE
52. fclose(stdin);
53. #endif // LOCAL\_JUDGE
54. **return** 0;
55. }

**舞蹈链**

**精确覆盖**

1. **const** **int** maxnode = 1001000;
2. **const** **int** maxn = 1010;
3. **struct** DLX{
4. **int** D[maxnode], U[maxnode], R[maxnode],L[maxnode];
5. **int** H[maxn], col[maxnode],row[maxnode];
6. **int** ans[maxn];
7. **int** size;
8. **int** n, m;
9. **int** ansed;
10. **void** init(**int** \_n, **int** \_m){
11. n = \_n;
12. m = \_m;
13. **for**(**int** i = 0; i <= m; ++i){
14. D[i] = U[i] = i;
15. R[i] = i + 1;
16. L[i] = i - 1;
17. col[i] = i;
18. row[i] = 0;
19. }
20. L[0] = m;
21. R[m] = 0;
22. memset(ans, 0, **sizeof**(ans));
23. memset(H, -1, **sizeof**(H));
24. size = m;
25. ansed = 0;
26. }
27. **void** push(**int** r, **int** c){
28. size ++;
29. D[size] = D[c];
30. U[size] = c;
31. U[D[c]] = size;
32. D[c] = size;
33. col[size] = c;
34. row[size] = r;
35. **if**(H[r] < 0){
36. H[r] = R[size] = L[size] = size;
37. }
38. **else**{
39. L[size] = H[r];
40. L[R[H[r]]] = size;
41. R[size] = R[H[r]];
42. R[H[r]] = size;
43. }
44. }
45. **void** del(**int** c){
46. L[R[c]] = L[c];
47. R[L[c]] = R[c];
48. **for**(**int** i = D[c]; i != c; i = D[i]){
49. **for**(**int** j = R[i]; j != i; j = R[j]){
50. D[U[j]] = D[j];
51. U[D[j]] = U[j];
52. }
53. }
54. }
55. **void** reback(**int** c){
56. **for**(**int** i = U[c]; i != c; i = U[i]){
57. **for**(**int** j = L[i]; j != i; j = L[j]){
58. D[U[j]] = j;
59. U[D[j]] = j;
60. }
61. }
62. L[R[c]] = c;
63. R[L[c]] = c;
64. }
65. **bool** dancing(**int** dep){
66. **if**(R[0] == 0){
67. ansed = dep;
68. **return** **true**;
69. }
70. **int** c = R[0];
71. del(c);
72. **for**(**int** i = D[c]; i != c; i = D[i]){
73. ans[dep] = row[i];
74. **for**(**int** j = R[i]; j != i; j = R[j])
75. del(col[j]);
76. **if**(dancing(dep+1)){
77. **return** **true**;
78. }
79. **for**(**int** j = L[i]; j != i; j = L[j]){
80. reback(col[j]);
81. }
82. }
83. **return** **false**;
84. }
85. }dlx;
86. **int** main(){
87. **int** n,m;
88. scanf("%d %d", &n,&m);
89. dlx.init(n,m);
90. **for**(**int** i = 1; i <= n; ++i){
91. **int** k;
92. **int** c;
93. scanf("%d", &k);
94. **for**(**int** j = 0; j < k; ++j){
95. scanf("%d",&c);
96. dlx.push(i, c);
97. }
98. }
99. **if**(dlx.dancing(0)){
100. printf("%d", dlx.ansed);
101. sort(dlx.ans,dlx.ans+dlx.ansed);
102. **for**(**int** i = 0; i < dlx.ansed; ++i){
103. printf(" %d", dlx.ans[i]);
104. }
105. }
106. **else**
107. printf("NO");
108. **return** 0;
109. }

**重复覆盖**

1. **const** **int** NUM = 100 \* 60;
2. **struct** SLX {
3. **int** U[NUM], D[NUM], L[NUM], R[NUM];//上下左右的邻居
4. **int** Col[NUM], Row[NUM];//每个元素对应的列行
5. **int** col[NUM], row[NUM];//每一列的元素个数，行的行首
6. **int** si;
7. **int** ans[1000], ansd;
8. **void** ini(**int** n, **int** m) {
9. **for** (**int** i = 0; i <= m; i++) {
10. R[i] = i + 1;
11. L[i] = i - 1;
12. D[i] = U[i] = i;
13. Col[i] = i;
14. Row[i] = 0;
15. col[i] = 0;
16. }
17. R[m] = 0;
18. L[0] = m;
19. si = m;
20. **for** (**int** i = 1; i <= n; i++) {
21. row[i] = -1;
22. }
23. }
24. **void** add(**int** r, **int** c) {
25. si++;
26. Col[si] = c;
27. Row[si] = r;
28. col[c]++;
29. D[si] = D[c];
30. U[D[c]] = si;
31. U[si] = c;
32. D[c] = si;
33. **if** (row[r] < 0) {
34. row[r] = R[si] = L[si] = si;
35. } **else** {
36. R[si] = R[row[r]];
37. L[R[row[r]]] = si;
38. L[si] = row[r];
39. R[row[r]] = si;
40. }
41. }
42. **void** remove(**int** c) {
43. **for** (**int** i = D[c]; i != c; i = D[i]) {
44. R[L[i]] = R[i];
45. L[R[i]] = L[i];
46. }
47. }
48. **void** resume(**int** c) {
49. **for** (**int** i = D[c]; i != c; i = D[i]) {
50. R[L[i]] = L[R[i]] = i;
51. }
52. }
53. **int** geth() { //最少还要处理几列
54. **int** ret = 0;
55. **bool** vis[80];
56. memset(vis, 0, **sizeof**(vis));
57. **for** (**int** c = R[0]; c; c = R[c]) {
58. **if** (!vis[c]) {
59. ret++;
60. **for** (**int** i = D[c]; i != c; i = D[i]) {
61. **for** (**int** j = R[i]; j != i; j = R[j]) {
62. vis[Col[j]] = **true**;
63. }
64. }
65. }
66. }
67. **return** ret;
68. }
69. **void** dance(**int** k) {
70. **if** (!R[0]) {
71. ansd = min(ansd, k);
72. **return**;
73. } **else** **if**(k+geth()>=ansd)
74. **return**;
75. **int** c = R[0];
76. **for** (**int** i = R[0]; i; i = R[i])
77. **if** (col[i] < col[c])
78. c = i;
79. **for** (**int** i = D[c]; i != c; i = D[i]) {
80. remove(i);
81. **for** (**int** j = R[i]; j != i; j = R[j]) {
82. remove(j);
83. col[Col[j]]--;
84. }
85. dance(k + 1);
86. **for** (**int** j = L[i]; j != i; j = L[j])
87. resume(j),col[Col[j]]++;
88. resume(i);
89. }
90. }
91. } dlx;

**单点替换、单点增减、区间求和、区间最值**

1. #define lson l , m , rt << 1
2. #define rson m + 1 , r , rt << 1 | 1
3. **const** **int** maxn = 222222;
5. **int** MAX[maxn<<2];
6. **int** MIN[maxn<<2];
7. **int** SUM[maxn<<2];
8. **int** max(**int** a,**int** b) {
9. **if**(a>b)**return** a;
10. **else** **return** b;
11. }
12. **int** min(**int** a,**int** b) {
13. **if**(a<b)**return** a;
14. **else** **return** b;
15. }
16. **void** PushUP(**int** rt) {
17. MAX[rt] = max(MAX[rt<<1], MAX[rt<<1|1]);
18. MIN[rt] = min(MIN[rt<<1], MIN[rt<<1|1]);
19. SUM[rt] = SUM[rt<<1] + SUM[rt<<1|1];
20. }
21. **void** build(**int** l,**int** r,**int** rt) {
22. **if** (l == r) {
23. scanf("%d",&MAX[rt]);
24. MIN[rt] = MAX[rt];
25. SUM[rt] = MAX[rt];
26. **return** ;
27. }
28. **int** m = (l + r) >> 1;
29. build(lson);
30. build(rson);
31. PushUP(rt);
32. }
33. **void** update(**int** p,**int** tihuan,**int** l,**int** r,**int** rt) {
34. **if** (l == r) {
35. MAX[rt] = tihuan;
36. MIN[rt] = tihuan;
37. SUM[rt] = tihuan;
38. **return** ;
39. }
40. **int** m = (l + r) >> 1;
41. **if** (p <= m) update(p, tihuan,lson);
42. **else** update(p, tihuan, rson);
43. PushUP(rt);
44. }
45. **void** update1(**int** p,**int** add,**int** l,**int** r,**int** rt) {
46. **if** (l == r) {
47. SUM[rt] = SUM[rt] + add;
48. **return** ;
49. }
50. **int** m = (l + r) >> 1;
51. **if** (p <= m) update1(p, add,lson);
52. **else** update1(p, add, rson);
53. PushUP(rt);
54. }
55. **int** query(**int** L,**int** R,**int** l,**int** r,**int** rt) {
56. **if** (L <= l && r <= R) {
57. **return** MAX[rt];
58. }
59. **int** m = (l + r) >> 1;
60. **int** ret = -1;
61. **if** (L <= m) ret = max(ret, query(L, R, lson));
62. **if** (R > m)  ret =  max(ret, query(L, R, rson));
63. **return** ret;
64. }
65. **int** query1(**int** L,**int** R,**int** l,**int** r,**int** rt) {
66. **if** (L <= l && r <= R) {
67. **return** MIN[rt];
68. }
69. **int** m = (l + r) >> 1;
70. **int** ret = 99999;
71. **if** (L <= m) ret = min(ret, query1(L, R, lson));
72. **if** (R > m)  ret =  min(ret, query1(L, R, rson));
73. **return** ret;
74. }
75. **int** queryhe(**int** L,**int** R,**int** l,**int** r,**int** rt) {
76. **if** (L <= l && r <= R) {
77. **return** SUM[rt];
78. }
79. **int** m = (l + r) >> 1;
80. **int** ret = 0;
81. **if** (L <= m) ret += queryhe(L, R, lson);
82. **if** (R > m)  ret +=  queryhe(L, R, rson);
83. **return** ret;
84. }
85. **int** main() {
86. **int** n, m;
87. **while** (~scanf("%d%d",&n,&m)) {
88. build(1, n, 1);
89. **while** (m --) {
90. **char** op[2];
91. **int** a, b;
92. scanf("%s%d%d",op,&a,&b);
93. **if** (op[0] == 'Q') { //区间求最大
94. printf("%d\n",query(a, b, 1, n, 1));
95. }
96. **else** **if**(op[0]=='U') //单点替换
97. update(a, b, 1, n, 1);
98. **else** **if**(op[0]=='M') { //区间求最小
99. printf("%d\n",query1(a, b, 1, n, 1));
100. }
101. **else** **if**(op[0]=='H') { //区间求和
102. printf("%d\n",queryhe(a, b, 1, n, 1));
103. }
104. **else** **if**(op[0]=='S') { //单点增加
105. scanf("%d%d",&a,&b);
106. update1(a, b, 1, n, 1);
107. }
108. **else** **if**(op[0]=='E') { //单点减少
109. scanf("%d%d",&a,&b);
110. update1(a, -b, 1, n, 1);
111. }
112. }
113. }
114. **return** 0;
115. }

**区间替换**

1. #define max(a,b) (a>b)?a:b
2. #define min(a,b) (a>b)?b:a
3. #define lson l , m , rt << 1
4. #define rson m + 1 , r , rt << 1 | 1
5. #define LL long long
6. **const** **int** maxn = 100100;
8. **int** lazy[maxn<<2];
9. **int** sum[maxn<<2];
11. **void** PushUp(**int** rt) { //由左孩子、右孩子向上更新父节点
12. sum[rt] = sum[rt<<1] + sum[rt<<1|1];
13. }
14. **void** PushDown(**int** rt,**int** m) { //向下更新
15. **if** (lazy[rt]) { //懒惰标记
16. lazy[rt<<1] = lazy[rt<<1|1] = lazy[rt];
17. sum[rt<<1] = (m - (m >> 1)) \* lazy[rt];
18. sum[rt<<1|1] = ((m >> 1)) \* lazy[rt];
19. lazy[rt] = 0;
20. }
21. }
22. **void** build(**int** l,**int** r,**int** rt) { //建树
23. lazy[rt] = 0;
24. **if** (l== r) {
25. scanf("%d",&sum[rt]);
26. **return** ;
27. }
28. **int** m = (l + r) >> 1;
29. build(lson);
30. build(rson);
31. PushUp(rt);
32. }
33. **void** update(**int** L,**int** R,**int** c,**int** l,**int** r,**int** rt) { //更新
34. **if** (L <= l && r <= R) {
35. lazy[rt] = c;
36. sum[rt] = c \* (r - l + 1);
37. **return** ;
38. }
39. PushDown(rt, r - l + 1);
40. **int** m = (l + r) >> 1;
41. **if** (L <= m) update(L, R, c, lson);
42. **if** (R > m) update(L, R, c, rson);
43. PushUp(rt);
44. }
45. LL query(**int** L,**int** R,**int** l,**int** r,**int** rt) {
46. **if** (L <= l && r <= R) {
47. **return** sum[rt];
48. }
49. PushDown(rt, r - l + 1);
50. **int** m = (l + r) >> 1;
51. LL ret = 0;
52. **if** (L <= m) ret += query(L, R, lson);
53. **if** (m < R) ret += query(L, R, rson);
54. **return** ret;
55. }
56. **int** main() {
57. **int**  n, m;
58. **char** str[5];
59. **while**(scanf("%d%d",&n,&m)) {
60. build(1, n, 1);
61. **while** (m--) {
62. scanf("%s",str);
63. **int** a, b, c;
64. **if**(str[0]=='T') {
65. scanf("%d%d%d",&a,&b,&c);
66. update(a, b, c, 1, n, 1);
67. } **else** **if**(str[0]=='Q') {
68. scanf("%d%d",&a,&b);
69. cout<<query(a,b,1,n,1)<<endl;
70. }
71. }
72. }
73. **return** 0;
74. }

**区间增减**

1. #define max(a,b) (a>b)?a:b
2. #define min(a,b) (a>b)?b:a
3. #define lson l , m , rt << 1
4. #define rson m + 1 , r , rt << 1 | 1
5. #define LL \_\_int64
6. **const** **int** maxn = 100100;
7. **using** **namespace** std;
8. LL lazy[maxn<<2];
9. LL sum[maxn<<2];
10. **void** putup(**int** rt) {
11. sum[rt] = sum[rt<<1] + sum[rt<<1|1];
12. }
13. **void** putdown(**int** rt,**int** m) {
14. **if** (lazy[rt]) {
15. lazy[rt<<1] += lazy[rt];
16. lazy[rt<<1|1] += lazy[rt];
17. sum[rt<<1] += lazy[rt] \* (m - (m >> 1));
18. sum[rt<<1|1] += lazy[rt] \* (m >> 1);
19. lazy[rt] = 0;
20. }
21. }
22. **void** build(**int** l,**int** r,**int** rt) {
23. lazy[rt] = 0;
24. **if** (l == r) {
25. scanf("%I64d",&sum[rt]);
26. **return** ;
27. }
28. **int** m = (l + r) >> 1;
29. build(lson);
30. build(rson);
31. putup(rt);
32. }
33. **void** update(**int** L,**int** R,**int** c,**int** l,**int** r,**int** rt) {
34. **if** (L <= l && r <= R) {
35. lazy[rt] += c;
36. sum[rt] += (LL)c \* (r - l + 1);
37. **return** ;
38. }
39. putdown(rt, r - l + 1);
40. **int** m = (l + r) >> 1;
41. **if** (L <= m) update(L, R, c, lson);
42. **if** (m < R) update(L, R, c, rson);
43. putup(rt);
44. }
45. LL query(**int** L,**int** R,**int** l,**int** r,**int** rt) {
46. **if** (L <= l && r <= R) {
47. **return** sum[rt];
48. }
49. putdown(rt, r - l + 1);
50. **int** m = (l + r) >> 1;
51. LL ret = 0;
52. **if** (L <= m) ret += query(L, R, lson);
53. **if** (m < R) ret += query(L, R, rson);
54. **return** ret;
55. }
56. **int** main() {
57. **int** n, m;
58. **int** a, b, c;
59. **char** str[5];
60. scanf("%d%d",&n,&m);
61. build(1, n, 1);
62. **while** (m--) {
63. scanf("%s",str);
64. **if** (str[0] == 'Q') {
65. scanf("%d%d",&a,&b);
66. printf("%I64d\n",query(a, b, 1, n, 1));
67. } **else** **if**(str[0]=='C') {
68. scanf("%d%d%d",&a,&b,&c);
69. update(a, b, c, 1, n, 1);
70. }
71. }
72. **return** 0;
73. }

**米勒拉宾**

1. **typedef** **long** **long** LL;
2. LL mulmod( LL a, LL b , LL p ){
3. LL  d =1;
4. a = a%p;
5. **while**( b>0 ){
6. **if**(b&1)     d = (d\*a)%p;
7. a = (a\*a)%p;
8. b>>=1;
9. }
10. **return** d;
11. }
12. **bool** witness( LL a,LL n){
13. LL d = n-1 ;
14. **if**( n ==2 ) **return** **true** ;
15. **if**( !(n&1) ) **return** **false** ;
16. **while**(!(d&1)) d = d/2;
17. LL t = mulmod(a,d,n);
18. **while**((d!=n-1) && (t!=1)&&(t!=n-1)){
19. t = mulmod( t ,2,n);
20. d=d<<1;
21. }
22. **return** (t==n-1)||(d&1);
23. }
25. **bool** isprime( LL n){
26. **int** a[3] = {2,7,61};
27. **for**(**int** i=0;i<3;i++)
28. **if**(!witness(a[i],n))
29. **return** **false**;
30. **return** **true**;
31. }
32. **int** main(){
33. LL s;
34. cin>>s;
35. **if**(isprime(s))
36. cout<<"YES";
37. **else**
38. cout<<"NO";
39. **return** 0;
40. }

**并查集**

1. **int** par[maxn];
2. **int** Find(**int** x){
3. **if**(par[x] == x)
4. **return** x;
5. **return** par[x] = Find(par[x]);
6. }
7. **void** Union(**int** x, **int** y){
8. par[Find(x)] = Find(y);
9. }
10. **void** init(){
11. **for**(**int** i = 0; i <= n; ++i)
12. par[i] = i;
13. }

**带权并查集**

1. **int** par[maxn];
2. ll d[maxn];
3. **int** ans;
5. **int** Find(**int** x){
6. **if**(x != par[x]){
7. **int** k = par[x];
8. par[x] = Find(par[x]);
9. d[x] += d[k];
10. }
11. **return** par[x];
12. }
13. **void** init(**int** n){
14. memset(d,0,**sizeof**(d));
15. **for**(**int** i = 0; i <= n; i++)
16. par[i] = i;
17. }
18. **void** Union(**int** x, **int** y, **int** v){
19. **int** xx = Find(x);
20. **int** yy = Find(y);
21. **if**(xx == yy)
22. **if**(d[x] + v != d[y])
23. ans++;
24. **else**{
25. par[yy] = xx;
26. d[yy] = d[x] - d[y]  + v;
27. }
28. }
29. **int** main(){
30. **int** n, m;
31. **while**(scanf("%d%d", &n, &m) != EOF){
32. ans = 0;
33. init(n);
34. **int** x, y, v;
35. **for**(**int** i = 0; i < m; i++){
36. scanf("%d%d%d", &x, &y, &v);
37. x -= 1;
38. Union(x, y, v);
39. }
40. printf("%d\n", ans);
41. }
42. **return** 0;
43. }

**最短路**

**Dijkstra**

1. **const** **int** inf = 0x3f3f3f3f;
2. **const** **int** maxn = 1010;
3. **int** G[maxn][maxn];
4. **int** vst[maxn];
5. **int** d[maxn];
6. **int** m, n;
7. **void** Dijkstra(**int** s){
8. **for**(**int** i = 1; i <= n; ++i){
9. d[i] = inf;
10. vst[i] = 0;
11. }
12. d[s] = 0;
13. vst[s] = 1;
14. **for**(**int** i = 0; i < n-1; ++i){
15. **int** Min = inf;
16. **int** k = s;
17. **for**(**int** j = 1; j <= n; ++j){
18. **if**(!vst[j] && Min > d[j]){
19. Min = d[j];
20. k = j;
21. }
22. }
23. vst[k] = 1;
24. **for**(**int** j = 1; j <= n; ++j){
25. **if**(G[k][j])
26. **if**(!vst[j] && d[j] > d[k]+G[k][j]){
27. d[j] = d[k] + G[k][j];
28. }
29. }
30. }
31. }
32. **int** main(){
33. **while**(~scanf("%d %d", &m, &n)){
34. memset(G, 0, **sizeof**(G));
35. **for**(**int** i = 0; i < m; ++i){
36. **int** x, y, v;
37. scanf("%d%d%d", &x, &y, &v);
38. **if**( G[x][y] == 0 || (G[x][y] && v < G[x][y]) ){
39. G[x][y] = v;
40. G[y][x] = v;
41. }
42. }
43. Dijkstra(1);
44. printf("%d\n", d[n]);
45. }
46. **return** 0;
47. }

**Spfa**

1. **const** **int** maxn = 1010;
2. **const** **int** inf = 0x3f3f3f3f;
3. **int** n, m;
4. **int** G[maxn][maxn];
5. **int** d[maxn];
6. **int** vst[maxn];
7. **void** spfa(**int** s){
8. queue<**int**> q;
9. **for**(**int** i = 0; i <= n; ++i){
10. d[i] = inf;
11. }
12. d[s] = 0;
13. memset(vst, 0, **sizeof**(vst));
14. vst[s] = 1;
15. q.push(s);
16. **while**(!q.empty()){
17. **int** k = q.front();
18. vst[k] = 0;
19. q.pop();
20. **for**(**int** j = 1; j <= n; ++j){
21. **if**(G[k][j])
22. **if**(d[j] > d[k]+G[k][j]){
23. d[j] = d[k] + G[k][j];
24. **if**(!vst[j]){
25. vst[j] = 1;
26. q.push(j);
27. }
28. }
29. }
30. }
31. }
32. **int** main(){
33. **while**(~scanf("%d %d", &m, &n)){
34. memset(G, 0, **sizeof**(G));
35. **for**(**int** i = 0; i < m; ++i){
36. **int** x, y, v;
37. scanf("%d %d %d", &x, &y, &v);
38. **if**(!G[x][y] || (G[x][y] && G[x][y] > v) ){
39. G[x][y] = G[y][x] = v;
40. }
41. }
42. spfa(1);
43. printf("%d\n", d[n]);
44. }
45. **return** 0;
46. }

**SPFA**

1. **const** **int** maxn = 210;
2. **const** **int** inf = 0x3f3f3f3f;
4. **int** n,m,s;
5. **double** v;
6. pair<**double**,**double**> G[maxn][maxn];
7. **int** vst[maxn];
8. **double** d[maxn];
9. **int** cot[maxn];//检测环路
11. **int** spfa(**int** s){
12. **for**(**int** i = 0; i <= n; ++i){
13. vst[i] = 0;
14. d[i] = 0;
15. cot[i] = 0;
16. }
17. vst[s] = 1;
18. cot[s] = 1;
19. d[s] = v;
20. queue<**int**> q;
21. q.push(s);
22. **while**(!q.empty()){
23. **int** k = q.front();
24. q.pop();
25. vst[k] = 0;
26. **for**(**int** j = 1; j <= n; ++j){
27. **if**(d[j]<(d[k]-G[k][j].second)\*G[k][j].first){
28. d[j]=(d[k]-G[k][j].second)\*G[k][j].first;
29. **if**(!vst[j]){
30. vst[j] = 1;
31. q.push(j);
32. cot[j] ++;
33. **if**(cot[j] > n)
34. **return** **true**;
35. }
36. }
37. }
38. }
39. **return** **false**;
40. }
41. **int** main(){
42. **while**(~scanf("%d%d%d%lf", &n,&m,&s,&v)){
43. memset(G, 0, **sizeof**(G));
44. **int** x, y;
45. **double** p,q;
46. **for**(**int** i = 0; i < m; ++i){
47. scanf("%d%d",&x,&y);
48. scanf("%lf%lf",&p,&q);
49. G[x][y] = make\_pair(p,q);
50. scanf("%lf%lf",&p,&q);
51. G[y][x] = make\_pair(p,q);
52. }
53. **if**(spfa(s)){
54. printf("YES\n");
55. }
56. **else**{
57. printf("NO\n");
58. }
59. }
60. **return** 0;
61. }

**SPFA 链式向前星**

1. **typedef** **long** **long** ll;
2. **const** **int** maxn = 1e6+10;
3. **const** **int** inf = 0x3f3f3f3f;
5. **struct** edge{
6. **int** to, w, next;
7. };
9. edge e[maxn];
10. **int** head[maxn];
11. **int** vst[maxn];
12. ll d[maxn];
13. **int** t, n, m;
14. **int** x[maxn], y[maxn], v[maxn];
16. **void** add(**int** i, **int** u, **int** v, **int** w){
17. e[i].to = v;
18. e[i].w = w;
19. e[i].next = head[u];
20. head[u] = i;
21. }
22. **void** spfa(**int** s){
23. **for**(**int** i = 0; i <= n; ++i){
24. vst[i] = 0;
25. d[i] = inf;
26. }
27. vst[s] = 1;
28. d[s] = 0;
29. queue<**int**> q;
30. q.push(s);
31. **while**(!q.empty()){
32. **int** k = q.front();
33. q.pop();
34. vst[k] = 0;
35. **for**(**int** j = head[k]; j != -1; j = e[j].next){
36. **if**(d[e[j].to] > d[k] + e[j].w){
37. d[e[j].to] = d[k] + e[j].w;
38. **if**(!vst[e[j].to]){
39. vst[e[j].to] = 1;
40. q.push(e[j].to);
41. }
42. }
43. }
44. }
45. }
46. **int** main(){
47. scanf("%d", &t);
48. **while**(t--){
49. scanf("%d%d", &n, &m);
50. memset(head, -1, **sizeof**(head));
51. memset(e, 0, **sizeof**(e));
52. **for**(**int** i = 0; i < m; ++i){
53. scanf("%d%d%d", &x[i], &y[i], &v[i]);
54. add(i, x[i], y[i], v[i]);
55. }
56. ll ans = 0;
57. spfa(1);
58. **for**(**int** i = 1; i <= n; ++i){
59. ans += d[i];
60. }
61. memset(head, -1, **sizeof**(head));
62. **for**(**int** i = 0; i < m; ++i){
63. add(i, y[i], x[i], v[i]);
64. }
65. spfa(1);
66. **for**(**int** i = 1; i <= n; ++i){
67. ans += d[i];
68. }
69. printf("%lld\n", ans);
70. }
71. **return** 0;
72. }

**二分法**

1. **const** **int** maxn=1e5+10;
2. **int** n,m,arr[maxn];
3. **bool** check(**int** x){
4. **int** sum=0,cot=1;
5. **for**(**int** i=0; i<n; ++i){
6. sum+=arr[i];
7. **if**(sum>x){
8. cot++;
9. sum=arr[i];
10. }
11. }
12. **return** (cot<=m)?1:0;
13. }
14. **int** main(){
15. cin>>n>>m;
16. **int** l,r=l=0;
17. **for**(**int** i=0; i<n; ++i){
18. cin>>arr[i];
19. r+=arr[i];
20. l=max(l,arr[i]);
21. }
22. **int** ans;
23. **while**(l<=r){
24. **int** mid=(l+r)>>1;
25. **if**(check(mid))
26. {
27. ans=mid;
28. r=mid-1;
29. }
30. **else**    l=mid+1;
31. }
32. cout<<ans;
33. }

**二分查找**

1. **int** Find(**int** a[],**int** l,**int** r,**int** key){
2. **int** m;
3. **while**(l<=r){
4. m=(l+r)>>1;
5. **if**(a[m]==key)   **return** m;
6. **else** **if**(key>a[m])   l=m+1;
7. **else**    r=m-1;
8. }
9. **return** -1;
10. }

**最小生成树**

**Kruskcal**

1. **const** **int** verNum = 30;
2. **const** **int** arcNum = 2400;
4. **typedef** **struct** node{
5. **int** val, start, end;
6. node(**int** s,**int** e,**int** v){
7. start = s;
8. end = e;
9. val = v;
10. }
11. node(){
12. ;
13. }
14. }Node;
16. **int** n, m;
17. Node v[arcNum];
18. **int** cot;
19. **int** par[arcNum];
20. **void** init(){
21. **for**(**int** i = 0; i <= m; ++i)
22. par[i] = i;
23. }
24. **bool** cmp(Node a, Node b){
25. **if**(a.val < b.val)
26. **return** **true**;
27. **return** **false**;
28. }
29. **int** Find(**int** x){
30. **if**(par[x] == x)
31. **return** x;
32. **return** par[x] = Find(par[x]);
33. }
34. **void** Union(**int** x, **int** y){
35. par[Find(x)] = Find(y);
36. }
37. **int** Kruskcal(**int** s){
38. **int** sum = 0;
39. **for**(**int** i = 0; i < m; ++i){
40. **if**(Find(v[i].start) != Find(v[i].end)){
41. Union(v[i].start, v[i].end);
42. sum += v[i].val;
43. }
44. }
45. **return** sum;
46. }
47. **int** main(){
48. **while**(~scanf("%d", &n) && n){
49. m = 0;
50. cot = 0;
51. **for**(**int** j = 0; j < n-1; ++j){
52. **char** ch[2];
53. **int** pre, k, p, cost;
54. scanf("%s %d",ch, &k);
55. pre = ch[0] - 'A';
56. m += k\*2;
57. **for**(**int** i = 0; i < k; ++i){
58. scanf("%s %d",ch, &cost);
59. p = ch[0] - 'A';
60. v[cot] = node(pre,p,cost);
61. cot ++;
62. v[cot] = node(p, pre, cost);
63. cot ++;
64. }
65. }
66. init();
67. sort(v, v+m, cmp);
68. **int** ans = Kruskcal(0);
69. printf("%d\n",ans);
70. }
71. }

**Prim**

1. **const** **int** verNum = 30;
2. **const** **int** arcNum = 80;
3. **const** **int** inf = 0x3f3f3f3f;
5. **int** n;
6. **int** G[verNum][verNum];
7. **int** vst[verNum];
8. **int** d[verNum];
9. **int** sum;
10. **void** Prim(**int** s){
11. memset(vst, 0, **sizeof**(vst));
12. vst[s] = 1;
13. **for**(**int** i = 0; i < n; ++i)
14. d[i] = G[s][i];
15. **for**(**int** i = 0; i < n-1; ++i){
16. **int** Min = inf;
17. **int** k = s;
18. **for**(**int** j = 0; j < n; ++j){
19. **if**(!vst[j] && Min > d[j]){
20. Min = d[j];
21. k = j;
22. }
23. }
24. vst[k] = 1;
25. sum += Min;
26. **for**(**int** j = 0; j < n; ++j){
27. **if**(!vst[j] && d[j] > G[k][j])
28. d[j] = G[k][j];
29. }
30. }
31. }
32. **int** main(){
33. **while**(~scanf("%d", &n) && n){
34. sum = 0;
35. **for**(**int** i = 0; i < n; ++i){
36. **for**(**int** j = 0; j < n; ++j)
37. G[i][j] = inf;
38. }
39. **for**(**int** i = 0; i < n-1; ++i){
40. **char** ch[2];
41. **int** pre, p, cost;
42. **int** k;
43. scanf("%s %d", ch, &k);
44. pre = ch[0] - 'A';
45. **for**(**int** j = 0; j < k; ++j){
46. scanf("%s %d", ch, &cost);
47. p = ch[0] - 'A';
48. G[pre][p] = G[p][pre] = cost;
49. }
50. }
51. Prim(0);
52. printf("%d\n", sum);
53. }
54. **return** 0;
55. }

**BFS**

1. **const** **int** maxn = 1e5+10;
2. **int** m, n;
3. //  1033 -8179 素数替换
4. **bool** prime[maxn];
5. **void** init(){
6. memset(prime, 1, **sizeof**(prime));
7. prime[1] = 0;
8. **for**(**int** i = 2; i < maxn; ++i){
9. **for**(**int** j = i\*2; j < maxn; j += i)
10. prime[j] = 0;
11. }
12. }
13. **int** pre[maxn];
14. **bool** vis[maxn];
15. **int** d[maxn];
16. **void** bfs(){
17. queue<**int**> q;
18. q.push(n);
19. vis[n] = 1;
20. **int** p;
21. **while**(!q.empty()){
22. p = q.front();
23. **int** t = p;
24. q.pop();
25. **int** a[4];
26. **if**(p == m)
27. **return** ;
28. **for**(**int** i = 0; i < 4; ++i){
29. a[i] = p%10;
30. p /= 10;
31. }
32. **for**(**int** i = 0; i < 10; ++i){
33. **for**(**int** k = 0; k < 4; ++k){
34. **int** tmp = t - (a[k] - i)\* pow(10,k);
35. **if**( tmp < 10000 && tmp >= 1000 && i != a[k] && !vis[tmp] && prime[tmp]){
36. q.push(tmp);
37. d[tmp] = d[t] + 1;
38. vis[tmp] = 1;
39. //                    pre[tmp] = t;
40. }
41. }
42. }
43. }
44. }
45. //void Print(int x){
46. //    stack<int> s;
47. //    while(pre[x]){
48. //        s.push(x);
49. //        x = pre[x];
50. //    }
51. //    while(!s.empty()){
52. //        printf("%d  ", s.top());
53. //        s.pop();
54. //    }
55. //}
56. **int** main(){
57. **int** t;
58. init();
59. scanf("%d", &t);
60. **while**(t--){
61. scanf("%d %d", &n, &m);
62. memset(d, 0, **sizeof**(d));
63. memset(vis, 0, **sizeof**(vis));
64. bfs();
65. cout<<d[m]<<endl;
66. //        Print(m);
67. }
68. **return** 0;
69. }

**DFS**

1. **int** m,n;
2. **int** b\_x,b\_y,e\_x,e\_y;
3. **int** row[100],col[100];
4. **int** k=1;
5. **int** step[][2]={{0,-1},{-1,0},{0,1},{1,0}};
6. **int** flag[100][100];
7. **int** mask=0;
8. **int** mp[100][100];
9. **int** OutBorder(**int** x,**int** y){
10. **if**(x<1||x>m||y<1||y>n)  **return** 1;
11. **return** 0;
12. }
13. **void** dfs(**int** x,**int** y){
14. **if**(x==e\_x && y==e\_y){
15. mask=1;
16. **for**(**int** i=0;i<k;i++)
17. printf(i==0?"(%d,%d)":"->(%d,%d)",row[i],col[i]);
18. printf("\n");
19. **return** ;
20. }
21. **for**(**int** i=0;i<4;i++){
22. **int** cur\_x=x+step[i][0];
23. **int** cur\_y=y+step[i][1];
24. **if**( !OutBorder(cur\_x,cur\_y) && !flag[cur\_x][cur\_y] && mp[cur\_x][cur\_y] ){
25. row[k]=cur\_x;
26. col[k]=cur\_y;
27. k++;
28. flag[cur\_x][cur\_y]=1;
29. dfs(cur\_x,cur\_y);
30. flag[cur\_x][cur\_y]=0;
31. k--;
32. }
33. }
34. }
35. **int** main()
36. {
37. cin>>m>>n;
38. **for**(**int** i=1;i<=m;i++){
39. **for**(**int** j=1;j<=n;j++)
40. scanf(" %d",&mp[i][j]);
41. }
42. memset(flag,0,**sizeof**(flag));
43. cin>>b\_x>>b\_y;
44. cin>>e\_x>>e\_y;
45. flag[1][1]=1;
46. row[0]=b\_x;
47. col[0]=b\_y;
48. dfs(b\_x,b\_y);
49. **if**(mask==0)
50. printf("-1\n");
51. }

**树状数组**

1. **typedef** **long** **long** ll;
2. **const** **int** maxn=1e6+5,N=1e5+5;
3. **int** Tree[maxn],a[N],n;
4. ll ans,s[N];
5. **inline** **int** lowbit(**int** x){
6. **return** (x&-x);
7. }
8. **void** add(**int** x,**int** val){
9. **for**(**int** i=x;i<=maxn;i+=lowbit(i))
10. Tree[i]+=val;
11. }
12. **int** get(**int** x){
13. **int** sum=0;
14. **for**(**int** i=x;i;i-=lowbit(i))
15. sum+=Tree[i];
16. **return** sum;
17. }
18. **int** main(){
19. **for**(**int** i = 1; i < N; ++i)
20. s[i]=s[i-1]+i;
21. scanf("%d",&n);
22. **for**(**int** i = 1; i <= n; ++i){//下标不能取0
23. scanf("%d",&a[i]);
24. add(i,a[i]);
25. }
26. **for**(**int** i = 1; i <= n; ++i)
27. printf("%4d", get(i));
28. }

**String**

**KMP**

1. **int**\* getNextArray(string s){
2. **int** \*next=**new** **int**[s.length()];
3. next[0] = -1;
4. **if**(s.length()==1)
5. **return** next;
6. next[1]=0;
7. **int** i=2,cn=0;
8. **while**(i<s.length()){
9. **if**(s[i-1]==s[cn])//cn 往前跳的值
10. next[i++] = ++cn;
11. **else** **if**(cn>0)//cn 继续往前跳
12. cn = next[cn];
13. **else**//cn 跳到头
14. next[i++] = 0;
15. }
16. **return** next;
17. }
18. **int** kmp(string s,string m){
19. **if**(s.length()<1||m.length()<1)
20. **return** -1;
21. **int** i=0,j=0;
22. **int** \*next=getNextArray(m);
23. **while**(i<s.length() && j<m.length()){
24. **if**(s[i]==m[j])
25. i++,j++;
26. **else** **if**(next[j]==-1)//开头配不上
27. i++;
28. **else**
29. j=next[j];
30. }
31. **return** j==m.length()?i-j:-1;
32. }
33. **int** main(){
34. string s,m;
35. cin>>s>>m;
36. cout<<kmp(s,m)<<endl;
37. }

**Manacher**

1. **char** \*manacherString(string s){
2. **char** \*res = **new** **char**[s.length() \* 2 + 1];
3. **int** index = 0;
4. **for** (**int** i = 0; i < s.length() \* 2 + 1;i++)
5. res[i] = (i & 1) ? s[index++] : '#';
6. **return** res;
7. }
8. **int** manacher(string s){
9. **if** (s.length() == 0)
10. **return** 0;
11. **char** \*charArr = manacherString(s);
12. **int** charArrLen = s.length() \* 2 + 1;
13. **int** \*pArr = **new** **int**[charArrLen];
14. **int** C = -1, R = -1;    //C为中心  R右为边界
15. **int** Max = INT\_MIN;
16. **for** (**int** i = 0; i < charArrLen ; i++){
17. pArr[i] = R > i ? min(pArr[2 \* C - i] , R-i) : 1;       //2、3种情况  1、4(边界)种情况
18. **while** (i + pArr[i] < charArrLen && i - pArr[i] > -1){   //判定是否越界
19. **if** (charArr[i + pArr[i]] == charArr[i - pArr[i]])  //暴力扩
20. pArr[i]++;
21. **else**
22. **break**;
23. }
24. **if** (i + pArr[i] > R){    //更新右边界
25. R = i + pArr[i];
26. C = i;
27. }
28. Max = max(Max , pArr[i]);
29. }
30. **return** Max - 1;//因增加#原因
31. }
32. **int** main(){
33. string s;
34. cin>>s;
35. cout<<manacher(s)<<endl;
36. }

**AC自动机**

1. // 求目标串中出现了几个模式串
2. **struct** Trie{
3. **int** next[500010][26], fail[500010], end[500010];
4. **int** root, L;
5. **int** newnode(){
6. **for** (**int** i = 0; i < 26; i++){
7. next[L][i] = -1;
8. }
9. end[L++] = 0;
10. **return** L - 1;
11. }
12. **void** init(){
13. L = 0;
14. root = newnode();
15. }
16. **void** insert(**char** buf[]){
17. **int** len = (**int**)strlen(buf);
18. **int** now = root;
19. **for** (**int** i = 0; i < len; i++){
20. **if** (next[now][buf[i] - 'a'] == -1){
21. next[now][buf[i] - 'a'] = newnode();
22. }
23. now = next[now][buf[i] - 'a'];
24. }
25. end[now]++;
26. }
27. **void** build(){
28. queue<**int**>Q;
29. fail[root] = root;
30. **for** (**int** i = 0; i < 26; i++){
31. **if** (next[root][i] == -1){
32. next[root][i] = root;
33. }
34. **else**{
35. fail[next[root][i]] = root;
36. Q.push(next[root][i]);
37. }
38. }
39. **while** (!Q.empty()){
40. **int** now = Q.front();
41. Q.pop();
42. **for** (**int** i = 0;i < 26;i++){
43. **if** (next[now][i] == -1){
44. next[now][i] = next[fail[now]][i];
45. }
46. **else**{
47. fail[next[now][i]]=next[fail[now]][i];
48. Q.push(next[now][i]);
49. }
50. }
51. }
52. }
53. **int** query(**char** buf[]){
54. **int** len = (**int**)strlen(buf);
55. **int** now = root;
56. **int** res = 0;
57. **for** (**int** i = 0; i < len; i++){
58. now = next[now][buf[i] - 'a'];
59. **int** temp = now;
60. **while** (temp != root){
61. res += end[temp];
62. end[temp] = 0;
63. temp = fail[temp];
64. }
65. }
66. **return** res;
67. }
68. };
69. **char** buf[1000010];
70. Trie ac;
71. **int** main(){
72. **int** T;
73. **int** n;
74. scanf("%d", &T);
75. **while**(T--){
76. scanf("%d", &n);
77. ac.init();
78. **for** (**int** i = 0; i < n; i++){
79. scanf("%s", buf);
80. ac.insert(buf);
81. }
82. ac.build();
83. scanf("%s", buf);
84. printf("%d\n", ac.query(buf));
85. }
86. **return** 0;
87. }

**主席树**

1. **typedef** **long** **long** LL;
2. #define lson l, m
3. #define rson m+1, r
4. **const** **int** N=60005;
5. **int** a[N], Hash[N];
6. **int** T[N], L[N<<5], R[N<<5], sum[N<<5];
7. **int** S[N];
8. **int** n, m, tot;
9. **struct** node{
10. **int** l, r, k;
11. **bool** Q;
12. }op[10005];
14. **int** build(**int** l, **int** r){
15. **int** rt=(++tot);
16. sum[rt]=0;
17. **if**(l!=r){
18. **int** m=(l+r)>>1;
19. L[rt]=build(lson);
20. R[rt]=build(rson);
21. }
22. **return** rt;
23. }
25. **int** update(**int** pre, **int** l, **int** r, **int** x, **int** val){
26. **int** rt=(++tot);
27. L[rt]=L[pre], R[rt]=R[pre], sum[rt]=sum[pre]+val;
28. **if**(l<r){
29. **int** m=(l+r)>>1;
30. **if**(x<=m)
31. L[rt]=update(L[pre], lson, x, val);
32. **else**
33. R[rt]=update(R[pre], rson, x, val);
34. }
35. **return** rt;
36. }
37. **int** lowbit(**int** x){
38. **return** x&(-x);
39. }
40. **int** use[N];
41. **void** add(**int** x, **int** pos, **int** val){
42. **while**(x<=n){
43. S[x]=update(S[x], 1, m, pos, val);
44. x+=lowbit(x);
45. }
46. }
48. **int** Sum(**int** x){
49. **int** ret=0;
50. **while**(x>0){
51. ret+=sum[L[use[x]]];
52. x-=lowbit(x);
53. }
54. **return** ret;
55. }
56. **int** query(**int** u, **int** v, **int** lr, **int** rr, **int** l, **int** r, **int** k){
57. **if**(l>=r)
58. **return** l;
59. **int** m=(l+r)>>1;
60. **int** tmp=Sum(v)-Sum(u)+sum[L[rr]]-sum[L[lr]];
61. **if**(tmp>=k){
62. **for**(**int** i=u;i;i-=lowbit(i))
63. use[i]=L[use[i]];
64. **for**(**int** i=v;i;i-=lowbit(i))
65. use[i]=L[use[i]];
66. **return** query(u, v, L[lr], L[rr], lson, k);
67. }
68. **else**{
69. **for**(**int** i=u;i;i-=lowbit(i))
70. use[i]=R[use[i]];
71. **for**(**int** i=v;i;i-=lowbit(i))
72. use[i]=R[use[i]];
73. **return** query(u, v, R[lr], R[rr], rson, k-tmp);
74. }
75. }
76. **void** modify(**int** x, **int** p, **int** d){
77. **while**(x<=n){
78. S[x]=update(S[x], 1, m, p, d);
79. x+=lowbit(x);
80. }
81. }
82. **int** main(){
83. **int** t;
84. scanf("%d", &t);
85. **while**(t--){
86. **int** q;//n个数 q次查询
87. scanf("%d%d", &n, &q);
88. tot=0;
89. m=0;
90. **for**(**int** i=1;i<=n;i++){
91. scanf("%d", &a[i]);
92. Hash[++m]=a[i];
93. }
94. **for**(**int** i=0;i<q;i++){
95. **char** s[10];
96. scanf("%s", s);
97. **if**(s[0]=='Q'){//查询[l,r]第k小的数
98. scanf("%d%d%d", &op[i].l, &op[i].r, &op[i].k);
99. op[i].Q=1;
100. }
101. **else**{//更改第i个数的值为r
102. scanf("%d%d", &op[i].l, &op[i].r);
103. op[i].Q=0;
104. Hash[++m]=op[i].r;
105. }
106. }
107. sort(Hash+1, Hash+1+m);
108. **int** mm=unique(Hash+1, Hash+1+m)-Hash-1;
109. m=mm;
110. T[0]=build(1, m);
111. **for**(**int** i=1;i<=n;i++)
112. T[i]=update(T[i-1], 1, m, lower\_bound(Hash+1, Hash+1+m, a[i])-Hash, 1);
113. **for**(**int** i=1;i<=n;i++)
114. S[i]=T[0];
115. **for**(**int** i=0;i<q;i++){
116. **if**(op[i].Q){
117. **for**(**int** j=op[i].l-1;j;j-=lowbit(j))
118. use[j]=S[j];
119. **for**(**int** j=op[i].r;j;j-=lowbit(j))
120. use[j]=S[j];
121. printf("%d\n", Hash[query(op[i].l-1, op[i].r, T[op[i].l-1], T[op[i].r], 1, m, op[i].k)]);
122. }
123. **else**{
124. modify(op[i].l, lower\_bound(Hash+1, Hash+1+m, a[op[i].l])-Hash, -1);
125. modify(op[i].l, lower\_bound(Hash+1, Hash+1+m, op[i].r)-Hash, 1);
126. a[op[i].l]=op[i].r;
127. }
128. }
129. }
130. **return** 0;
131. }

**ST表**

1. //预处理复杂度同为O(nlogn),查询时间上，ST表为O(1),线段树为O(nlogn)
2. **const** **int** MAXN = 1e5 + 10;
3. **int** Max[MAXN][21];
4. **int** Min[MAXN][21];
5. **int** Query(**int** l,**int** r) {
6. **int** k=log2(r-l+1);
7. **return** max(Max[l][k],Max[r-(1<<k)+1][k]);//把拆出来的区间分别取最值
8. }
9. **int** lQuery(**int** l,**int** r) {
10. **int** k=log2(r-l+1);
11. **return** min(Min[l][k],Min[r-(1<<k)+1][k]);//把拆出来的区间分别取最值
12. }
13. **int** main() {
14. **int** n, m;
15. scanf("%d %d", &n, &m);
16. **for**(**int** i = 1; i <= n; i++) {
17. scanf("%d", &Max[i][0]);
18. Min[i][0] = Max[i][0];
19. }
20. **for**(**int** j = 1; (1 << j) <= n; j++)
21. **for**(**int** i = 1; i + (1 << j) - 1 <= n; i++) {
22. Max[i][j] = max(Max[i][j-1], Max[i + (1 << (j - 1))][j - 1]);
23. Min[i][j] = min(Min[i][j-1], Min[i + (1 << (j - 1))][j - 1]);
24. }
25. **for**(**int** i = 0; i < m; i++) {
26. **int** l, r;
27. scanf("%d %d", &l, &r);
28. **int** maxItem = Query(l,r);
29. **int** minItem = lQuery(l,r);
30. printf("maxItem:");
31. printf("%4d\n", maxItem);
32. printf("minItem");
33. printf("%4d\n", minItem);
34. }
35. }

**矩阵快速幂**

1. **int** n,Mod;
2. **const** **int** maxn=5, maxm=5;
3. **struct** Matrix{
4. **int** n,m;
5. **int** a[maxn][maxm];
6. **void** clear(){
7. n=m=0;
8. memset(a,0,**sizeof**(a));
9. }
10. Matrix operator \*(**const** Matrix &b) **const**{
11. Matrix tmp;
12. tmp.clear();
13. tmp.n=n;tmp.m=b.m;
14. **for**(**int** i=0;i<n;i++)
15. **for**(**int** j=0;j<b.m;j++)
16. **for**(**int** k=0;k<m;k++)
17. tmp.a[i][j]+=a[i][k]\*b.a[k][j];
18. **return** tmp;
19. }
20. };
21. // [0, 1, 0, 0]    f[n-4]  f[n-3]
22. // [0, 0, 1, 0]    f[n-3]  f[n-2]
23. // [0, 0, 0, 1]    f[n-2]  f[n-1]
24. // [-1,1, 5, 1]    f[n-1]  f[n]
25. **void** Pow(**int** m){
26. Matrix s;
27. s.clear();
28. s.n=s.m=4;
29. s.a[3][3]=1;s.a[3][2]=5;
30. s.a[3][1]=1;s.a[3][0]=-1;
31. s.a[1][2]=1;s.a[2][3]=1;
32. s.a[0][1]=1;
33. Matrix ans;
34. ans.clear();
35. ans.n=4;
36. ans.m=1;
37. ans.a[0][0]=1;
38. ans.a[1][0]=5;
39. ans.a[2][0]=11;
40. ans.a[3][0]=36;
41. **while**(m){
42. **if**(m&1)
43. ans=s\*ans;
44. s=s\*s;
45. m>>=1;
46. }
47. cout<<ans.a[3][0]<<endl;
48. }
49. **int** main(){
50. **while**(cin>>n>>Mod){
51. **if**(!n)  **return** 0;
52. **if**(n<4){
53. **switch**(n){
54. **case** 1:
55. cout<<1%Mod<<endl;
56. **break**;
57. **case** 2:
58. cout<<5%Mod<<endl;
59. **break**;
60. **case** 3:
61. cout<<11%Mod<<endl;
62. **break**;
63. }
64. **continue**;
65. }
66. Pow(n-4);
67. }
68. **return** 0;
69. }

**快速幂**

1. **long** ksm(**long** x, **long** n, **long** MOD){
2. **long** ans = 1;
3. **while** (n > 0){
4. **if** ((n & 1) != 0)
5. ans = ans \* x % MOD;
6. x = x \* x % MOD;
7. n >>= 1;
8. }
9. **return** ans;
10. }

**欧拉函数O(sqrt(n))**

1. **long** euler(**long** n) {
2. **long** res = n, a = n;
3. **for** (**int** i = 2; i \* i <= n; ++i) {
4. **if** (a % i == 0) {
5. res = res / i \* (i - 1);
6. **while** (a % i == 0)
7. a /= i;
8. }
9. }
10. **if** (a > 1)
11. res = res / a \* (a - 1);
12. **return** res;
13. }

**欧拉降幂公式(A^x)%p = (A^(x%phi(p)+phi))%p**

1. **long** eulerDP(**long** A, BigInteger x, **long** p) {
2. **long** phi = euler(p);
3. **long** n = x.mod(BigInteger.valueOf(phi)).add(BigInteger.valueOf(phi)).longValue();
4. **return** ksm(A, n, p);
5. }

**递推求1-n欧拉函数 O(nlogn)**

1. final **static** **int** maxn = 1000001;
2. **static** **long**[] euler = **new** **long**[maxn];
3. **void** Init() {
4. euler[1] = 1;
5. **for** (**int** i = 2; i < maxn; ++i)
6. euler[i] = i;
7. **for** (**int** i = 2; i < maxn; ++i)
8. **if** (euler[i] == i)
9. **for** (**int** j = i; j < maxn; j += i)
10. euler[j] = euler[j] / i \* (i - 1);
11. }

**大整数快速幂**

1. **const** **int** mod=1e9+7;
2. **long** **long** quick\_mod(**long** **long** a,**long** **long** b){
3. **long** **long** ans=1;
4. **while**(b){
5. **if**(b&1){
6. ans=(ans\*a)%mod;
7. b--;
8. }
9. b/=2;
10. a=a\*a%mod;
11. }
12. **return** ans;
13. }//内部也用快速幂
14. **long** **long** quickmod(**long** **long** a,**char** \*b,**int** len){
15. **long** **long** ans=1;
16. **while**(len>0){
17. **if**(b[len-1]!='0'){
18. **int** s=b[len-1]-'0';
19. ans=ans\*quick\_mod(a,s)%mod;
20. }
21. a=quick\_mod(a,10)%mod;
22. len--;
23. }
24. **return** ans;
25. }
26. **int** main(){
27. **char** s[100050];
28. **int** a;      //求a^s%mod
29. **while**(~scanf("%d",&a)){
30. scanf("%s",s);
31. **int** len=strlen(s);
32. printf("%I64d\n",quickmod(a,s,len));
33. }
34. **return** 0;
35. }

**Gxgcd**

1. //x，y初始为任意值，最后变为一组特解
2. **int** exgcd(**int** a, **int** b, **int** &x, **int** &y) {
3. //对应最终情况，a=gcd(a,b),b=0,此时x=1，y为任意数
4. **if**(b == 0) {
5. x = 1;
6. y = 0;
7. **return** a;
8. }
9. //先递归到最终情况，再反推出初始情况
10. **int** r = exgcd(b, a % b, x, y);
11. **int** t = x; x = y; y = t - a / b \* y;
12. **return** r;     //gcd(a,b)
13. }

**扩展欧几里得求逆元**

1. **int** inv1(**int** a,**int** mod){
2. **int** x,y;
3. **int** d=exgcd(a,mod,x,y);
4. **if**(d==1) **return** (x%mod+mod)%mod;
5. **return** -1;
6. }

**费马小定理**

1. **int** inv2(**int** a,**int** mod){
2. **return** ksm(a,mod-2,mod);
3. }

**线性递推求1~mod的所以逆元**

1. **void** inv3(**int** mod){
2. inv[1]=1;
3. **for**(**int** i=2;i<=mod-1;i++){
4. inv[i]=(mod-mod/i)\*inv[mod%i]%mod;
5. cout<<inv[i]<<" ";
6. }
7. }

**线性素数筛**

1. **bool** check[100005];
2. **int** prime[100005];//储存第i个素数
3. **void** getprime(**int** n){
4. memset(check, 0, **sizeof**(check));
5. **int** tot = 0;// tot 初始为 0，用来记录质数总个数
6. **for** (**int** i = 2; i <= n; ++i){
7. **if** (!check[i])
8. prime[++tot] = i;
9. **for** (**int** j = 1; j <= tot; ++j){
10. **if** (i \* prime[j] > n)
11. **break**;
12. check[i \* prime[j]] = 1;
13. **if** (i % prime[j] == 0){
14. **break**;
15. }
16. }
17. }
18. }

**逆序数**

1. **const** **int** N=100050;
2. **int** n;
3. **long** **long** c[N];//c[n]表示a[1~n]的和，a数组省略
4. **struct** node{
5. **int** val,pos;
6. } a[100005];
7. **int** lowbit(**int** x){//求2^k
8. **return** x & -x;
9. }
10. **long** **long** getsum(**int** n){//区间查询，求a[1~n]的和
11. **long** **long** res = 0;
12. **while**(n>0){
13. res+=c[n];
14. n=n-lowbit(n);
15. }
16. **return** res;
17. }
18. **int** change(**int** x){//单点更新，将c[x]的值加1
19. **while**(x<=n){
20. c[x]++;
21. x+=lowbit(x);
22. }
23. }
24. **bool** cmp(node a,node b){//包含相同数
25. **if**(a.val!=b.val)
26. **return** a.val>b.val;
27. **return** a.pos>b.pos;
28. }
29. **int** main(){
30. **while**(cin>>n){
31. memset(c,0,**sizeof**(c));
32. **for**(**int** i=1; i<=n; i++){
33. cin>>a[i].val;
34. a[i].pos=i;
35. }
36. sort(a+1,a+n+1,cmp);
37. **long** **long** cnt=0;
38. **for**(**int** i=1; i<=n; i++){
39. change(a[i].pos); //修改最大数位置的值
40. cnt+=getsum(a[i].pos-1);
41. //最大数位置之前的所有位置和，即区间求和，可知比当前数小的数有多少个
42. }
43. cout<<cnt<<endl;
44. }
45. **return** 0;
46. }

**求排列组合数**

**语法：result=P(long n,long m); / result=long C(long n,long m);**

**参数 m：排列组合的上系数n：排列组合的下系数**

**返回值：排列组合数**

**注意： 符合数学规则：m<＝n**

1. **long** P(**long** n,**long** m){
2. **long** p=1;
3. **while**(m!=0){
4. p\*=n;
5. n--;
6. m--;
7. }
8. **return** p;
9. }
10. **long** C(**long** n,**long** m){
11. **long** i,c=1;
12. i=m;
13. **while**(i!=0){
14. c\*=n;n--;i--;
15. }
16. **while**(m!=0){
17. c/=m;m--;
18. }
19. **return** c;
20. }

**求某一天星期几**

**语法：result=weekday(int N,int M,int d)**

**参数：N,M,d：年月日，例如：2003,11,4**

**返回值：0：星期天，1星期一……**

**注意： 需要math.h**

**适用于1582年10月15日之后, 因为罗马教皇格里高利十三世在这一天启用新历法.**

1. **int** weekday(**int** N,**int** M,**int** d){
2. **int** m,n,c,y,w;
3. m=(M-2)%12;
4. **if** (M>=3) n=N;**else** n=N-1;
5. c=n/100;
6. y=n%100;
7. w=(**int**)(d+floor(13\*m/5)+y+floor(y/4)+floor(c/4)-2\*c)%7;
8. **while**(w<0) w+=7;
9. **return** w;
10. }

**任意进制转换**

**语法：conversion(char s1[],char s2[],char t[]);**

**参数：**

**s[]：转换前的数字 s2[]：转换后的数字 s2高位位数越大**

**d1：原进制数 d2：需要转换到的进制数**

1. **void** conversion(**char** s[],**char** s2[],**long** d1,**long** d2){
2. **long** i,j,t,num;
3. **char** c;
4. num=0;
5. **for** (i=0; s[i]!='\0'; i++){
6. **if** (s[i]<='9'&&s[i]>='0') t=s[i]-'0';
7. **else** t=s[i]-'A'+10;
8. num=num\*d1+t;
9. }
10. i=0;
11. **while**(1){
12. t=num%d2;
13. **if** (t<=9) s2[i]=t+'0';
14. **else** s2[i]=t+'A'-10;
15. num/=d2;
16. **if** (num==0) **break**;
17. i++;
18. }
19. **for** (j=0; j<i/2; j++){
20. c=s2[j];
21. s2[j]=s[i-j];
22. s2[i-j]=c;
23. }
24. s2[i+1]='\0';
25. }

**Ronberg算法计算积分**

**语法：result=integral(double a,double b);**

**参数：a：积分上限 b：积分下限**

**function f：积分函数**

**返回值：f在（a,b）之间的积分值**

**注意： function f(x)需要自行修改，程序中用的是sina(x)/x**

**需要 math.h 默认精度要求是1e-5**

1. **double** f(**double** x){
2. **return** sin(x)/x; //在这里插入被积函数
3. }
4. **double** integral(**double** a,**double** b){
5. **double** h=b-a;
6. **double** t1=(1+f(b))\*h/2.0;
7. **int** k=1;
8. **double** r1,r2,s1,s2,c1,c2,t2;
9. loop:
10. **double** s=0.0;
11. **double** x=a+h/2.0;
12. **while**(x<b){
13. s+=f(x);
14. x+=h;
15. }
16. t2=(t1+h\*s)/2.0;
17. s2=t2+(t2-t1)/3.0;
18. **if**(k==1){
19. k++;
20. h/=2.0;
21. t1=t2;
22. s1=s2;
23. **goto** loop;
24. }
25. c2=s2+(s2-s1)/15.0;
26. **if**(k==2){
27. c1=c2;
28. k++;
29. h/=2.0;
30. t1=t2;
31. s1=s2;
32. **goto** loop;
33. }
34. r2=c2+(c2-c1)/63.0;
35. **if**(k==3){
36. r1=r2;
37. c1=c2;
38. k++;
39. h/=2.0;
40. t1=t2;
41. s1=s2;
42. **goto** loop;
43. }
44. **while**(fabs(1-r1/r2)>1e-5){
45. r1=r2;
46. c1=c2;
47. k++;
48. h/=2.0;
49. t1=t2;
50. s1=s2;
51. **goto** loop;
52. }
53. **return** r2;
54. }

**行列式计算**

**语法：result=js(int s[][],int n)**

**参数：s[][]：行列式存储数组**

**n：行列式维数，递归用**

**返回值：行列式值**

**注意： 函数中常数N为行列式维度，需自行定**

1. #define N 3
2. **int** js(**int** s[][N],**int** n){
3. **int** z,j,k,r,total=0;
4. **int** b[N][N];
5. /\*b[N][N]用于存放，在矩阵s[N][N]中元素s[0]的余子式\*/
6. **if**(n>2){
7. **for**(z=0; z<n; z++){
8. **for**(j=0; j<n-1; j++)
9. **for**(k=0; k<n-1; k++)
10. **if**(k>=z) b[j][k]=s[j+1][k+1];
11. **else**
12. b[j][k]=s[j+1][k];
13. **if**(z%2==0) r=s[0][z]\*js(b,n-1); /\*递归调用\*/
14. **else** r=(-1)\*s[0][z]\*js(b,n-1);
15. total=total+r;
16. }
17. }
18. **else** **if**(n==2)
19. total=s[0][0]\*s[1][1]-s[0][1]\*s[1][0];
20. **return** total;
21. }

**杨辉三角**

**语法: void gen()**

**预置: const int Max;**

**注意: (Max一般不能超过22,否则要用高精度计算) int inta[Max][Max];**

**结果:inta为杨辉三角序列.inta[1][1]=1;inta[2][1]=1;inta[2][2]=1…**

1. **const** **int** Max = 22;
2. **int** inta[Max][Max];
3. **void** gen() {
4. **int** i,j;
5. **for**( i =1 ; i <= Max ; i++) {
6. inta[i][1] = 1 ;
7. inta[i][i] = 1 ;
8. }
9. inta[2][2] = 1   ;
10. **for**( i = 3 ; i<=Max ; i++) {
11. **for**(j =2 ; j<i ; j++ ) {
12. inta[i][j] = inta[i-1][j-1] + inta[i-1][j] ;
13. }
14. }
15. }

**字符串**

**数学**

**求三角形面积**

**语法：result=area3(float x1,float y1,float x2,float y2,float x3,float y3);**

**参数：x1～3：三角形3个顶点x坐标**

**y1～3：三角形3个顶点y坐标**

**返回值：三角形面积**

**注意： 需要 math.h**

1. **float** area3(**float** x1,**float** y1,**float** x2,**float** y2,**float** x3,**float** y3) {
2. **float** a,b,c,p,s;
3. a=sqrt((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2));
4. b=sqrt((x1-x3)\*(x1-x3)+(y1-y3)\*(y1-y3));
5. c=sqrt((x3-x2)\*(x3-x2)+(y3-y2)\*(y3-y2));
6. p=(a+b+c)/2;
7. s=sqrt(p\*(p-a)\*(p-b)\*(p-c));
8. **return** s;
9. }

**两矢量间角度**

**语法：result=angle(double x1, double y1, double x2, double y2);**

**参数：x/y1～2：两矢量的坐标**

**返回值：两的角度矢量**

**注意： 返回角度为弧度制，并且以逆时针方向为正方向 需要 math.h**

1. #define PI 3.1415926
2. **double** angle(**double** x1, **double** y1, **double** x2, **double** y2) {
3. **double** dtheta,theta1,theta2;
4. theta1 = atan2(y1,x1);
5. theta2 = atan2(y2,x2);
6. dtheta = theta2 - theta1;
7. **while** (dtheta > PI)
8. dtheta -= PI\*2;
9. **while** (dtheta < -PI)
10. dtheta += PI\*2;
11. **return**(dtheta);
12. }

**语法：result=distance\_2d(float x1,float x2,float y1,float y2);**

**参数：x/y/z1～2：各点的x、y、z坐标**

**返回值：两点之间的距离**

**注意： 需要 math.h**

1. **float** distance\_2d(**float** x1,**float** x2,**float** y1,**float** y2) {
2. **return**(sqrt((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)));
3. }

6. **float** distance\_3d(**float** x1,**float** x2,**float** y1,**float** y2,**float** z1,**float** z2) {
7. **return**(sqrt((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)+(z1-z2)\*(z1-z2)));
8. }

**判断点是否在线段上**

**语法：result=Pointonline(Point p1,Point p2,Point p);**

**参数：p1、p2：线段的两个端点 p：被判断点**

**返回值：0：点在不在线段上；1：点在线段上**

**注意：若p线段端点上返回1 需要 math.h**

1. #define MIN(x,y) (x < y ? x : y)
2. #define MAX(x,y) (x > y ? x : y)
3. **typedef** **struct** {
4. **double** x,y;
5. } Point;
6. **int** FC(**double** x1,**double** x2) {
7. **if** (x1-x2<0.000002&&x1-x2>-0.000002) **return** 1;
8. **else** **return** 0;
9. }
10. **int** Pointonline(Point p1,Point p2,Point p) {
11. **double** x1,y1,x2,y2;
12. x1=p.x-p1.x;
13. x2=p2.x-p1.x;
14. y1=p.y-p1.y;
15. y2=p2.y-p1.y;
16. **if** (FC(x1\*y2-x2\*y1,0)==0) **return** 0;
17. **if** ((MIN(p1.x,p2.x)<=p.x&&p.x<=MAX(p1.x,p2.x))&&
18. (MIN(p1.y,p2.y)<=p.y&&p.y<=MAX(p1.y,p2.y)))
19. **return** 1;
20. **else** **return** 0;
21. }

**判断两线段是否相交**

**语法：result=lineintersect(Point p1,Point p2,Point p3,Point p4);**

**参数：p1～4：两条线段的四个端点**

**返回值：0：两线段不相交；1：两线段相交；2两线段首尾相接**

**注意： p1!=p2;p3!=p4;**

1. #define MIN(x,y) (x < y ? x : y)
2. #define MAX(x,y) (x > y ? x : y)
3. **typedef** **struct** {
4. **double** x,y;
5. } Point;
6. **int** lineintersect(Point p1,Point p2,Point p3,Point p4) {
7. Point tp1,tp2,tp3;
8. **if**((p1.x==p3.x&&p1.y==p3.y)||(p1.x==p4.x&&p1.y==p4.y)||(p2.x==p3.x&&p2.y==p3.y)||(p2.x==p4.x&&p2.y==p4.y))
9. **return** 2;
10. //快速排斥试验
11. **if**((MIN(p1.x,p2.x)<=p3.x&&p3.x<=MAX(p1.x,p2.x)&&MIN(p1.y,p2.y)<=p3.y&&p3.y<=MAX(p1.y,p2.y))||
12. (MIN(p1.x,p2.x)<=p4.x&&p4.x<=MAX(p1.x,p2.x)&&MIN(p1.y,p2.y)<=p4.y&&p4.y<=MAX(p1.y,p2.y)))
13. ;
14. **else** **return** 0;
15. //跨立试验
16. tp1.x=p1.x-p3.x;
17. tp1.y=p1.y-p3.y;
18. tp2.x=p4.x-p3.x;
19. tp2.y=p4.y-p3.y;
20. tp3.x=p2.x-p3.x;
21. tp3.y=p2.y-p3.y;
22. **if** ((tp1.x\*tp2.y-tp1.y\*tp2.x)\*(tp2.x\*tp3.y-tp2.y\*tp3.x)>=0) **return** 1;
23. **else** **return** 0;
24. }

**判断线段与直线是否相交**

**语法：result=lineintersect(Point p1,Point p2,Point p3,Point p4);**

**参数：p1、p2：线段的两个端点 p3、p4：直线上的两个点**

**返回值：0：线段直线不相交；1：线段和直线相交**

**注意： 如线段在直线上，返回 1**

1. **typedef** **struct** {
2. **double** x,y;
3. } Point;
4. **int** lineintersect(Point p1,Point p2,Point p3,Point p4) {
5. Point tp1,tp2,tp3;
6. tp1.x=p1.x-p3.x;
7. tp1.y=p1.y-p3.y;
8. tp2.x=p4.x-p3.x;
9. tp2.y=p4.y-p3.y;
10. tp3.x=p2.x-p3.x;
11. tp3.y=p2.y-p3.y;
12. **if** ((tp1.x\*tp2.y-tp1.y\*tp2.x)\*(tp2.x\*tp3.y-tp2.y\*tp3.x)>=0) **return** 1;
13. **else** **return** 0;
14. }

**点到线段最短距离**

**语法：result=mindistance(Point p1,Point p2,Point q);**

**参数：p1、p2：线段的两个端点 q：判断点**

**返回值：点q到线段p1p2的距离**

**注意： 需要 math.h**

1. #define MIN(x,y) (x < y ? x : y)
2. #define MAX(x,y) (x > y ? x : y)
3. **typedef** **struct** {
4. **double** x,y;
5. } Point;
6. **double** mindistance(Point p1,Point p2,Point q) {
7. **int** flag=1;
8. **double** k;
9. Point s;
10. **if** (p1.x==p2.x) {
11. s.x=p1.x;
12. s.y=q.y;
13. flag=0;
14. }
15. **if** (p1.y==p2.y) {
16. s.x=q.x;
17. s.y=p1.y;
18. flag=0;
19. }
20. **if** (flag) {
21. k=(p2.y-p1.y)/(p2.x-p1.x);
22. s.x=(k\*k\*p1.x+k\*(q.y-p1.y)+q.x)/(k\*k+1);
23. s.y=k\*(s.x-p1.x)+p1.y;
24. }
25. **if** (MIN(p1.x,p2.x)<=s.x&&s.x<=MAX(p1.x,p2.x))
26. **return** sqrt((q.x-s.x)\*(q.x-s.x)+(q.y-s.y)\*(q.y-s.y));
27. **else**
28. **return**
29. MIN(sqrt((q.x-p1.x)\*(q.x-p1.x)+(q.y-p1.y)\*(q.y-p1.y)),sqrt((q.x-p2.x)\*(q.x-p2.x)+(q.y-p2.y)\*(q.y-p2.y)));
30. }

**求两直线的交点**

**语法：result=linecorss (Point p1,Point p2,Point q);**

**参数：p1～p4：直线上不相同的两点 \*p：通过指针返回结果**

**返回值：1：两直线相交；2：两直线平行**

**注意： 如需要判断两线段交点，检验k和对应k1（注释中）的值是否在0～1之间，用在0～1之间的那个求交点**

1. **typedef** **struct** {
2. **double** x,y;
3. } Point;
4. **int** linecorss(Point p1,Point p2,Point p3,Point p4,Point \*p) {
5. **double** k;
6. **if** ((p4.y-p3.y)\*(p2.x-p1.x)-(p4.x-p3.x)\*(p2.y-p1.y)==0) **return** 0;
7. **if** ((p4.x-p3.x)\*(p1.y-p3.y)-(p4.y-p3.y)\*(p1.x-p3.x)==0&&
8. (p2.x-p1.x)\*(p1.y-p3.y)-(p2.y-p1.y)\*(p1.x-p3.x)==0) **return** 0;
10. k=((p4.x-p3.x)\*(p1.y-p3.y)-(p4.y-p3.y)\*(p1.x-p3.x))/((p4.y-p3.y)\*(p2.x-p1.x)-(p4.x-p3.x)\*(p2.y-p1.y));
11. (\*p).x=p1.x+k\*(p2.x-p1.x);
12. (\*p).y=p1.y+k\*(p2.y-p1.y);
13. **return** 1;
14. }

**求两条线段的交点**

**语法：Result=IntersectPoint (Point p1,Point p2,Point p3,Point p4,Point &p);**

**参数：P1～P4：两条线断4个端点 P：线段交点**

**返回值：如果两条线段平行无交点，返回 0，否则返回 1**

1. **struct** Point {
2. **float** x,y;
3. };
4. **int** IntersectPoint (Point p1,Point p2,Point p3,Point p4,Point &p) {
5. **float** a,b,c,d,e,f;
6. a=p2.y-p1.y;
7. b=p1.x-p2.x;
8. c=p1.y\*(p2.x-p1.x)+p1.x\*(p2.y-p1.y);
9. d=p4.y-p3.y;
10. e=p3.x-p4.x;
11. f=p3.y\*(p4.x-p3.x)+p1.x\*(p4.y-p3.y);
12. **if** (a\*e==b\*d)
13. **return** 0;
14. **else** {
15. p.x=(e\*c-b\*f)/(b\*d-a\*e);
16. p.y=(d\*c-a\*f)/(a\*e-b\*d);
17. **return** 1;
18. }
19. }

**数论**

**x的二进制长度**

**语法：result=BitLength(int x);**

**参数：x：测长的x**

**返回值：x的二进制长度**

1. **int** BitLength(**int** x) {
2. **int** d = 0;
3. **while** (x > 0) {
4. x >>= 1;
5. d++;
6. }
7. **return** d;
8. }

**返回x的二进制表示中从低到高的第i位**

**语法：result=BitAt(int x, int i);**

**参数：x：十进制 x i：要求二进制的第i位**

**返回值：返回x的二进制表示中从低到高的第i位**

**注意： 最低位为第一位**

1. **int** BitAt(**int** x, **int** i) {
2. **return** ( x & (1 << (i-1)) );
3. }

**取幂运算**

**语法：result=Modular\_Expoent(int a,int b,int n);**

**参数：a、b、n：a^b mod n 的对应参数**

**返回值：a^b mod n 的值**

**注意： 需要BitLength和BitAt**

1. **int** Modular\_Expoent(**int** a,**int** b,**int** n) {
2. **int** i, y=1;
3. **for** (i = BitLength(b); i > 0; i--) {
4. y = (y\*y)%n;
5. **if** (BitAt(b,i) > 0)
6. y = (y\*a)%n;
7. }
8. **return** y;
9. }

**求解模线性方程**

**语法：result＝modular\_equation(int a,int b,int n);**

**参数：a、b、n：ax=b (mod n) 的对应参数**

**返回值：方程的解**

1. **int** ext\_euclid(**int** a,**int** b,**int** &x,**int** &y) { //求gcd(a,b)=ax+by
2. **int** t,d;
3. **if** (b==0) {
4. x=1;
5. y=0;
6. **return** a;
7. }
8. d=ext\_euclid(b,a %b,x,y);
9. t=x;
10. x=y;
11. y=t-a/b\*y;
12. **return** d;
13. }
15. **void** modular\_equation(**int** a,**int** b,**int** n) {
16. **int** e,i,d;
17. **int** x,y;
18. d=ext\_euclid(a,n,x,y);
19. **if** (b%d>0)
20. printf("No answer!\n");
21. **else** {
22. e=(x\*(b/d))%n;
23. **for** (i=0; i<d; i++)
24. printf("The %dth answer is : %ld\n",i+1,(e+i\*(n/d))%n);
25. }
26. }

**求解模线性方程组(中国余数定理)**

**语法：result=Modular\_Expoent(int a,int b,int n);**

**参数：B[]、W[]：a=B[] (mod W[]) 的对应参数**

**返回值：a 的值**

**注意： 其中W[],B[]已知，W[i]>0且W[i]与W[j]互质, 求a**

1. **int** ext\_euclid(**int** a,**int** b,**int** &x,**int** &y) { //求gcd(a,b)=ax+by
2. **int** t,d;
3. **if** (b==0) {
4. x=1;
5. y=0;
6. **return** a;
7. }
8. d=ext\_euclid(b,a %b,x,y);
9. t=x;
10. x=y;
11. y=t-a/b\*y;
12. **return** d;
13. }
14. **int** China(**int** B[],**int** W[],**int** k) {
15. **int** i;
16. **int** d,x,y,a=0,m,n=1;
17. **for** (i=0; i<k; i++)
18. n\*=W[i];
19. **for** (i=0; i<k; i++) {
20. m=n/W[i];
21. d=ext\_euclid(W[i],m,x,y);
22. a=(a+y\*m\*B[i])%n;
23. }
24. **if** (a>0) **return** a;
25. **else** **return**(a+n);
26. }

**筛法素数产生器**

**语法：result=prime(int a[],int n);**

**参数：a[]：用于返回素数的数组**

**n：产生n以内的素数，按升序放入a[]中**

**返回值：n以内素数的个数**

1. **int** prime(**int** a[],**int** n) {
2. **int** i,j,k,x,num,\*b;
3. n++;
4. n/=2;
5. b=**new** **int**[(n+1)\*2];
6. a[0]=2;
7. a[1]=3;
8. num=2;
9. **for**(i=1; i<=2\*n; i++)
10. b[i]=0;
11. **for**(i=3; i<=n; i+=3)
12. **for**(j=0; j<2; j++) {
13. x=2\*(i+j)-1;
14. **while**(b[x]==0) {
15. a[num++]=x;
16. **for**(k=x; k<=2\*n; k+=x)
17. b[k]=1;
18. }
19. }
20. **return** num;
21. }

**求一个数每一位相加之和**

**语法：result=digadd(int n)**

**参数：n：待求数字**

**返回值：各数字之和**

1. **int** digadd(**int** n) {
2. **int** i=0,k=0;
3. **while**(i=n%10,n/=10)  k+=i;
4. **return** k+i;
5. }

**质因数分解**

**语法：result=int reduce(int prime[],int pn,int n,int rest[])**

**参数：prime[]：素数表，至少需要达到sqrt(n)**

**pn：素数表的元素个数**

**n：待分解的数**

**rest：分解结果，按照升序排列**

1. **int** reduce(**int** prime[],**int** pn,**int** n,**int** rest[]) {
2. **int** i,k=0;
3. **for**(i=0; i<pn; i++) {
4. **if** (n==1) **break**;
5. **if** (prime[i]\*prime[i]>n) {
6. rest[k++]=n;
7. **break**;
8. }
9. **while**(n%prime[i]==0) {
10. n/=prime[i];
11. rest[k++]=prime[i];
12. }
13. }
14. **return** k;
15. }

**高斯消元法解线性方程组**

1. **const** **int** MAXN=50;
2. **int** a[MAXN][MAXN];//增广矩阵
3. **int** x[MAXN];//解集
4. **bool** free\_x[MAXN];//标记是否是不确定的变元
5. **inline** **int** gcd(**int** a,**int** b) {
6. **int** t;
7. **while**(b!=0) {
8. t=b;
9. b=a%b;
10. a=t;
11. }
12. **return** a;
13. }
14. **inline** **int** lcm(**int** a,**int** b) {
15. **return** a/gcd(a,b)\*b;//先除后乘防溢出
16. }
17. // 高斯消元法解方程组(Gauss-Jordan elimination).(-2表示有浮点数解，但无整数解，
18. //-1表示无解，0表示唯一解，大于0表示无穷解，并返回自由变元的个数)
19. //有equ个方程，var个变元。增广矩阵行数为equ,分别为0到equ-1,列数为var+1,分别为0到var.
20. **int** Gauss(**int** equ,**int** var) {
21. **int** i,j,k;
22. **int** max\_r;// 当前这列绝对值最大的行.
23. **int** col;//当前处理的列
24. **int** ta,tb;
25. **int** LCM;
26. **int** temp;
27. **int** free\_x\_num;
28. **int** free\_index;
29. **for**(**int** i=0; i<=var; i++) {
30. x[i]=0;
31. free\_x[i]=**true**;
32. }
33. //转换为阶梯阵.
34. col=0; // 当前处理的列
35. **for**(k = 0; k < equ && col < var; k++,col++) {
36. // 枚举当前处理的行.
37. // 找到该col列元素绝对值最大的那行与第k行交换.(为了在除法时减小误差)
38. max\_r=k;
39. **for**(i=k+1; i<equ; i++) {
40. **if**(abs(a[i][col])>abs(a[max\_r][col])) max\_r=i;
41. }
42. **if**(max\_r!=k) {// 与第k行交换.
43. **for**(j=k; j<var+1; j++) swap(a[k][j],a[max\_r][j]);
44. }
45. **if**(a[k][col]==0) {// 说明该col列第k行以下全是0了，则处理当前行的下一列.
46. k--;
47. **continue**;
48. }
49. **for**(i=k+1; i<equ; i++){// 枚举要删去的行.
50. **if**(a[i][col]!=0) {
51. LCM = lcm(abs(a[i][col]),abs(a[k][col]));
52. ta = LCM/abs(a[i][col]);
53. tb = LCM/abs(a[k][col]);
54. **if**(a[i][col]\*a[k][col]<0)tb=-tb;//异号的情况是相加
55. **for**(j=col; j<var+1; j++) {
56. a[i][j] = a[i][j]\*ta-a[k][j]\*tb;
57. }
58. }
59. }
60. }
61. // 1. 无解的情况: 化简的增广阵中存在(0, 0, ..., a)这样的行(a != 0).
62. **for** (i = k; i < equ; i++) {
63. // 对于无穷解来说，如果要判断哪些是自由变元，那么初等行变换中的交换就会影响，则要记录交换.
64. **if** (a[i][col] != 0) **return** -1;
65. }
66. // 2. 无穷解的情况: 在var \* (var + 1)的增广阵中出现(0, 0, ..., 0)这样的行，即说明没有形成严格的上三角阵.
67. // 且出现的行数即为自由变元的个数.
68. **if** (k < var) {
69. // 首先，自由变元有var - k个，即不确定的变元至少有var - k个.
70. **for** (i = k - 1; i >= 0; i--) {
71. // 第i行一定不会是(0, 0, ..., 0)的情况，因为这样的行是在第k行到第equ行.
72. // 同样，第i行一定不会是(0, 0, ..., a), a != 0的情况，这样的无解的.
73. free\_x\_num = 0; // 用于判断该行中的不确定的变元的个数，如果超过1个，则无法求解，它们仍然为不确定的变元.
74. **for** (j = 0; j < var; j++) {
75. **if** (a[i][j] != 0 && free\_x[j]) free\_x\_num++, free\_index = j;
76. }
77. **if** (free\_x\_num > 1) **continue**; // 无法求解出确定的变元.
78. // 说明就只有一个不确定的变元free\_index，那么可以求解出该变元，且该变元是确定的.
79. temp = a[i][var];
80. **for** (j = 0; j < var; j++) {
81. **if** (a[i][j] != 0 && j != free\_index) temp -= a[i][j] \* x[j];
82. }
83. x[free\_index] = temp / a[i][free\_index]; // 求出该变元.
84. free\_x[free\_index] = 0; // 该变元是确定的.
85. }
86. **return** var - k; // 自由变元有var - k个.
87. }
88. // 3. 唯一解的情况: 在var \* (var + 1)的增广阵中形成严格的上三角阵.
89. // 计算出Xn-1, Xn-2 ... X0.
90. **for** (i = var - 1; i >= 0; i--) {
91. temp = a[i][var];
92. **for** (j = i + 1; j < var; j++) {
93. **if** (a[i][j] != 0) temp -= a[i][j] \* x[j];    //--因为x[i]存的是temp/a[i][i]的值，即是a[i][i]=1时x[i]对应的值
94. }
95. **if** (temp % a[i][i] != 0) **return** -2; // 说明有浮点数解，但无整数解.
96. x[i] = temp / a[i][i];
97. }
98. **return** 0;
99. }
100. **int** main(**void**) {
101. **int** i, j;
102. **int** equ,var;//equ \* (var + 1)  参数
103. **while** (scanf("%d %d", &equ, &var) != EOF) {
104. memset(a, 0, **sizeof**(a));
105. **for** (i = 0; i < equ; i++) {
106. **for** (j = 0; j < var + 1; j++) {
107. scanf("%d", &a[i][j]);
108. }
109. }
110. **int** free\_num = Gauss(equ,var);
111. **if** (free\_num == -1) printf("无解!\n");
112. **else** **if** (free\_num == -2) printf("有浮点数解，无整数解!\n");
113. **else** **if** (free\_num > 0) {
114. printf("无穷多解! 自由变元个数为%d\n", free\_num);
115. **for** (i = 0; i < var; i++) {
116. **if** (free\_x[i]) printf("x%d 是不确定的\n", i + 1);
117. **else** printf("x%d: %d\n", i + 1, x[i]);
118. }
119. } **else** {
120. **for** (i = 0; i < var; i++) {
121. printf("x%d: %d\n", i + 1, x[i]);
122. }
123. }
124. printf("\n");
125. }
126. **return** 0;
127. }

**dp**

**01 背包**

**有n 种不同的物品，每个物品有两个属性，size 体积，value 价值，**

**现在给一个容量为 w 的背包，问最多可带走多少价值的物品。**

1. **for** (**int** i=0; i<n; i++)
2. **for** (**int** j=w; j>=size[i]; j--)
3. f[j] = max(f[j], f[j-size[i]]+value[i]);

**完全背包**

1. **for** (**int** i=0; i<n; i++)
2. **for** (**int** j=size[i]; j<=w; j++)
3. f[j] = max(f[j], f[j-size[i]]+value[i]);

**最长公共子序列**

1. **int** LCS2(string s1,string s2){//s1待匹配串 s2匹配串
2. **int** len1=s1.length();
3. **int** len2=s2.length();
5. **int** leftabove,left,above;
6. **int** \*dp=**new** **int** [len2+1];
7. memset(dp,0,(len2+1)\***sizeof**(**int**));
8. **for**(**int** i=1;i<=len1;i++){
9. leftabove=left=dp[0];
10. above=dp[1];
11. **for**(**int** j=1;j<=len2;j++){
12. **if**(s1[i-1]==s2[j-1])   dp[j]=leftabove+1;
13. **else**    dp[j]=max(left,above);
14. leftabove=above;
15. above=dp[j+1];
16. left=dp[j];
17. }
18. }
19. cout<<dp[len2]<<endl;
20. }
21. **int** main() {
22. string s1, s2;
23. **while**(cin>>s1>>s2){
24. LCS2(s1,s2);
25. }
26. }

**STL**

1. vector<**int**> v;
2. v.begin();   //容器的起始位置
3. v.end();    //容器最后一个位置后的位置
4. v.front();v.back();   //返回第一个元素（最后一个元素，但不判断时候存在
5. v.empty();    //返回是否容器为空
6. v.clear();    //清空容器
7. v.erase(m);    //删除m位置的数据，并返回下一个数据的地址（m是迭代器）
8. v.erase(m,n);     //删除m到n之间的数据，并返回下一个数据的地址
9. v2.assign(8,1);   // 重新给vec2赋值，8个成员的初始值都为1
10. v.push\_back(element);    //压入一个元素到末端
11. v.pop\_back();    //弹出最后一个元素
12. v.reserve(100);v.resize(101);    //resize已经创建空间如果再v.push\_back();空间就会到101，而reserve只是预留空间并没有真正创建，v.push\_back();只是在第1位
13. v.size();v.capacity();       //size表示的是已经创建的空间大小也可以表示元素个数可用v[]的形式直接访问，capacity容器容量，是预留空间并没有实际创建
14. swap(a,b);      //交换两个元素的位置如:swap(v[0],v[1]);
15. vector<**int**>  v(10);    //创建一个前十个元素为int的容器
16. vector<string> v(10,string("I"));  //使容器的前10个元素都为string型，并且都初始化为I
17. vector<string> v1(v2);    //对于已经存在的v2创建一个v1副本
18. v.insert(place,element);
19. v.insert(place,n,element);     //在place（迭代器）位插入n个元素
20. //注：对vector元素的访问可以用类似c语言的v[],但是最好用v.at(),它会检查是否越界更安全
21. v[0];    // A
22. v.at[0];  // B 这样越界的时候比较安全

**reverse(vec.begin(),vec.end());**

**deque双端队列**

1. #include<queue>
2. deque <**int**> deq;//初始化对象为空
3. deque <**int**> deq1(10,6);//对象初始化有10个值为6的元素
4. deque <**int**>:: iterator it;
5. **for**(it = deq.begin(); it!=deq.end(); it++){
6. cout<<\*it<<endl;
7. }
8. deq.push\_back(ele);//从队列尾部插入
9. deq.push\_front(ele);//从队列头部插入
10. deq.insert(deq.begin()+1,3,9);//从队列中间插入三个9
11. //和vector一样，双向队列也可以用下标的形式访问，也可以用at
12. deq.at(n);//返回的是n这个下标的值
13. //也可以直接deq[n]
15. deq1.at(1)=10;
16. deq1[2]=12;
17. //从deq1序列的前后各移去一个元素
18. deq1.pop\_front();
19. deq1.pop\_back();
21. deq.erase(deq.begin()+1);//清除deq的第二个元素
23. //对deq2赋值并显示
24. deq2.assign(8,1);
25. cout<<"deq2.assign(8,1):"<<endl;
26. put\_deque(deq2,"deq2");
27. //erase(),assign()是大多数容器都有的操作

**集合set（特点是没有重复元素且元素时有序的）**

**注：set是STL中一种标准关联容器（vector,list,string,deque都是序列容器，而set，multiset，map，multimap是标准关联容器），它底层使用平衡的搜索树——红黑树实现，插入删除操作时仅仅需要指针操作节点即可完成，不涉及到内存移动和拷贝，所以效率比较高。**

1. set<**int**> s;
2. s.insert(element);//插入元素
3. //遍历整个几何（不重复的有序序列）
4. set<**int**>:: iterator it;
5. **for**(it = s.begin(); it!=s.end(); it++){
6. cout<<(\*it)<<endl;
7. }
8. s.size();//返回元素个数
9. s.find(ele);//返回元素的下标，没找到的话返回s.end();
10. //获得两个set的并
11. set<**int**> s1;
12. set<**int**> s2;
13. set<**int**> s3;//存结果
14. set\_union(s1.begin(),s1.end(),s2.begin(),s2.end(),insert\_iterator<set<**int**> >(s3,s3.begin()));
15. //输出也可以用下面的形式
16. copy(s3.begin(),s3.end(),ostream\_iterator<**int**>(cout," "));
17. cout<<endl;
18. //获得两个set的交，注意进行集合操作之前接收结果的set要调用clear()函数清空一下
19. s3.clear();
20. set\_intersection(s1.begin(),s1.end(),s2.begin(),s2.end(),insert\_iterator<set<**int**> >(s3.s3.begin()));
21. copy(s3.begin(),s3.end(),ostream\_iterator<**int**>(cout," "));
22. cout<<endl;
24. //获得两个set的差
25. s3.clear();
26. set\_difference(s1.begin(),s1.end(),s2.begin(),s2.end(),insert\_iterator<set<**int**> >(s3,s3.begin()));
27. cout<<"Difference:";
28. copy(s3.begin(),s3.end(),ostream\_iterator<**int**>(cout," "));
29. cout<<endl;
30. //获得两个set的对称差，也就是假设两个集合分别为A和B那么对称差为AUB-A∩B
31. eg3.clear();
32. set\_symmetric\_difference(s1.begin(),s1.end(),s2.begin(),s2.end(),insert\_iterator<set<**int**> >(s3,s3.begin()));
33. copy(s3.begin(),s3.end(),ostream\_iterator<**int**>(cout," "));
34. cout<<endl;

**特别注意：常用的是将set升序排列**

1. set<**int**,less<**int**> > set1;//降序排列
2. set<**int**,greater<**int**> > set1;//升序排列

**map 和 Multimap映射**

1. #include<map>
3. map<**int**, string> mym;
4. //三种插入方法
5. mym.insert(pair<**int**,string>(120,"haha"));
6. mym.insert(map<**int**,string>::value\_type(312,"gaga"));
7. mym[120] = "haha";
8. //find()函数返回一个迭代器指向键值为key的元素，如果没找到就返回指向map尾部的迭代器。
9. map<**int** ,string >::iterator l\_it;
10. l\_it=maplive.find(112);//返回的是一个指针
11. **if**(l\_it==maplive.end())
12. cout<<"we do not find112"<<endl;
13. **else**
14. cout<<"wo find112"<<endl;
16. //也可以用这种方式来查找
17. map<**int** ,string> m;
18. m[1] = "haa";
19. m[44] = "xixi";
20. **if**(m[12]=="") puts("NO");
22. //删除一个元素
23. map<**int**,string> m;
24. map<**int**,string>::iterator it;
25. it = m.find(112);
26. **if**(it == m.end()){ exit;}
27. **else** m.erase(it);
28. //类似的也可以通过赋值为空来将一个元素删除
30. //Map中的swap不是一个容器中的元素交换，而是两个容器交换
32. begin() //返回指向map头部的迭代器
33. clear()// 删除所有元素
34. count()// 返回指定元素出现的次数
35. empty() //如果map为空则返回true
36. end() //返回指向map末尾的迭代器
37. equal\_range()// 返回特殊条目的迭代器对
38. erase() //删除一个元素
39. find() //查找一个元素
40. insert() //插入元素
41. lower\_bound() //返回键值>=给定元素的第一个位置
42. max\_size() //返回可以容纳的最大元素个数
43. rbegin() //返回一个指向map尾部的逆向迭代器
44. rend() //返回一个指向map头部的逆向迭代器
45. size() //返回map中元素的个数
46. swap()// 交换两个map
47. upper\_bound()// 返回键值>给定元素的第一个位置
48. value\_comp() //返回比较元素value的函数
49. //multimap允许重复的键值插入容器
50. //        \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
51. //        \* pair只包含一对数值:pair<int,char>                       \*
52. //        \* map是一个集合类型，永远保持排好序的，                   \*
53. //  pair  \* map每一个成员就是一个pair,例如：map<int,char>           \*
54. //        \* map的insert()可以把一个pair对象作为map的参数,例如map<p> \*
55. //        \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
56. #pragma warning(disable:4786)
57. #include<map>
58. #include<iostream>
59. **using** **namespace** std;
61. **int** main(**void**)
62. {
63. multimap<**int**,**char**\*> m;
64. //multimap的插入只能用insert()不能用数组
65. m.insert(pair<**int**,**char**\*>(1,"apple"));
66. m.insert(pair<**int**,**char**\*>(1,"pear"));//apple和pear的价钱完全有可能是一样的
67. m.insert(pair<**int**,**char**\*>(2,"banana"));
68. //multimap的遍历只能用迭代器方式不能用数组
69. cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;
70. multimap<**int**,**char**\*>::iterator i,iend;
71. iend=m.end();
72. **for**(i=m.begin();i!=iend;i++)
73. {
74. cout<<(\*i).second<<"的价钱是"
75. <<(\*i).first<<"元/斤\n";
76. }
77. cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;
78. //元素的反相遍历
79. multimap<**int**,**char**\*>::reverse\_iterator j,jend;
80. jend=m.rend();
81. **for**(j=m.rbegin();j!=jend;j++)
82. {
83. cout<<(\*j).second<<"的价钱是"
84. <<(\*j).first<<"元/斤\n";
85. }
86. cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;
87. //元素的搜索find(),pair<iterator,iterator>equal\_range(const key\_type &k)const
88. //和multiset的用法一样
89. multimap<**int**,**char**\*>::iterator s;
90. s=m.find(1);//find()只要找到一个就行了，然后立即返回。
91. cout<<(\*s).second<<"    "
92. <<(\*s).first<<endl;
93. cout<<"键值等于1的元素个数是："<<m.count(1)<<endl;
94. cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;
95. //删除 erase(),clear()
96. m.erase(1);
97. **for**(i=m.begin();i!=iend;i++)
98. {
99. cout<<(\*i).second<<"的价钱是"
100. <<(\*i).first<<"元/斤\n";
101. }
102. **return** 0;
103. }

**queue与priority\_queue**

1. #include<queue>
2. queue<**int**> q1;
3. q.push(x);//将元素插入到队列末尾
4. q.pop();//弹出队列的第一个元素（队首），并不会返回元素的值
5. q.front();//访问队首的元素
6. q.back();//访问队尾元素
7. q.size();//返回队列中元素的个数
8. priority\_queue<**int** >q1;
9. priority\_queue<pair<**int**,**int**> >q2;
10. priority\_queue<**int**,vector<**int**>,greater<**int**> >q3;//定义小的先出队

**stack 栈**

1. #include<stack>
2. stack<**int**> s;
3. s.push();//压栈
4. s.pop();//出栈
5. s.top();//获取栈顶元素
6. s.empty();
7. s.size();
8. s.end();
9. s.begin();

下面介绍一下stl中的一些常用的算法

1. find(); count(); unique(); sort();
2. copy(v.begin(),v.end(),l.begin());
3. //将v中的元素复制到l中。

**堆操作**

1. make\_heap(v.begin(),v.end());//创建堆
2. sort\_heap(v.begin(),v.end());//堆排序
3. push\_heap(v.begin(),v.end());//堆入队
4. pop\_heap(v.begin(),v.end());//堆出队

卡特兰 (Catalan) 数列 原理

令h(1)＝1，catalan数满足递归式：

h(n)= h(1)\*h(n-1) + h(2)\*h(n-2) + ... + h(n-1)h(1) (其中n>=2)

该递推关系的解为：h(n)=c(2n-2,n-1)/n (n=1,2,3,...)

1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452, …

1.括号化问题。

矩阵链乘： P=a1×a2×a3×……×an，依据乘法结合律，不改变其顺序，只用括号表示成对的乘积，试问有几种括号化的方案？(h(n)种)

2.出栈次序问题。

一个栈(无穷大)的进栈序列为1,2,3,..n,有多少个不同的出栈序列?

类似：有2n个人排成一行进入剧场。入场费5元。其中只有n个人有一张5元钞票，另外n人只有10元钞票，剧院无其它钞票，问有多少中方法使得只要有10元的人买票，售票处就有5元的钞票找零？(将持5元者到达视作将5元入栈，持10元者到达视作使栈中某5元出栈)

3.将多边行划分为三角形问题。

将一个凸多边形区域分成三角形区域的方法数?

类似：一位大城市的律师在她住所以北n个街区和以东n个街区处工作。每天她走2n个街区去上班。如果他

从不穿越（但可以碰到）从家到办公室的对角线，那么有多少条可能的道路？

类似：在圆上选择2n个点,将这些点成对连接起来使得所得到的n条线段不相交的方法数?

博弈

1.巴什博奕

问题模型：只有一堆n个物品，两个人轮流从这堆物品中取物品，规定每次至少取一个，最多取m个，最后取光者得胜。）

结论:n%(m+1)==0先手必败.否则先手必胜

变形:条件不变，改为最后取光的人输。 结论：（n-1）%(m+1)==0 先手必败,否则必胜

2. 威佐夫博奕

问题模型：有两堆各若干个物品，两个人轮流从某一堆或同时从两堆中取同样多的物品，规定每次至少取一个，多者不限，

最后取光者得胜。

结论: k=(b-a) (a>b) s = (double )( k \* ( sqrt(5.0) + 1 ) / 2 ); s==a 先手必败,否则必胜

3.尼姆博弈

问题模型：有三堆各若干个物品，两个人轮流从某一堆取任意多的物品，规定每次至少取一个，多者不限，最后取光者得胜。

结论: 当石子堆数为n堆时，则推广为当对每堆的数目进行亦或之后值为零是必败态。

4.sg函数

5.阶梯博弈

博弈在一列阶梯上进行...每个阶梯上放着自然数个点..两个人进行阶梯博弈...每一步则是将一个集体上的

若干个点( >=1 )移到前面去..最后没有点可以移动的人输..

阶梯博弈也是可以转化成尼姆博弈的.

把所有奇数阶梯看成N堆石子..做nim..把石子从奇数堆移动到偶数堆可以理解为拿走石子..就相当于几个

奇数堆的石子在做Nim..

6.Chomp!博弈(巧克力游戏)

有一个n\*m的棋盘，每次可以取走一个方格并拿掉它右边和上面的所有方格。拿到左下角的格子(1,1)者输，如下图是8\*3的

棋盘中拿掉(6,2)和(2,3)后的状态。

结论：答案是除了1\*1的棋盘，对于其他大小的棋盘，先手总能赢。

分析：有一个很巧妙的证明可以保证先手存在必胜策略，可惜这个证明不是构造性的，也就是说没有给出先手怎么下才能赢。

证明如下：

如果后手能赢，也就是说后手有必胜策略，使得无论先手第一次取哪个石子，后手都能获得最后的胜利。那么现在假设先手

取最右上角的石子(n,m)，接下来后手通过某种取法使得自己进入必胜的局面。但事实上，先手在第一次取的时候就可以和

后手这次取的一样，进入必胜局面了，与假设矛盾。

巧克力游戏的变形：

约数游戏：有1~n个数字，两个人轮流选择一个数字，并把它和它的约数擦去。擦去最后一个数的人赢，问谁会获胜。

分析：类似巧克力游戏，得到结论就是无论n是几，都是先手必胜。

翻棋子游戏：

题意：一个棋盘上每个格子有一个棋子，每次操作可以随便选一个朝上的棋子(x,y)，代表第i行第j列的棋子，选择一个形

如(x,b)或(a,y)(其中b < y，a < x)的棋子，然后把它和(x,y)一起翻转，无法操作的人输。

分析：把坐标为(x,y)的棋子看成大小分别为x和y的两堆石子，则本题转化为了经典的Nim游戏，如果难以把棋子看作石

子，可以先把Nim游戏中的一堆石子看成一个正整数，则Nim游戏中的每次操作是把其中一个正整数减小或者删除。

**运行时间计算**

1. 注：vector是顺序容器，没有find函数、#include<time.h>
2. **clock\_t** start,finish;
3. start=clock();
4. finish=clock();
5. cout<< (**double**)(finish - start) / CLOCKS\_PER\_SEC;//运行时间:

待完善：

网络流

Dp