

Contents

IIUM cat-us-trophy



.vimrc	1
template.cpp	1
Combinatorics	2
Data structures	3
String algorithms (SuffixA, Aho-Corasick, KMP)	4
Graph algorithms (LCA, SCC, NetFlow, MinCost, BPM)	7
Geometry	11
Appendices (ASCII table)	12

.vimrc

```
set ai ts=4 sw=4 st=4 noet nu nohls
syntax enable
filetype plugin indent on
map <F6> :w<CR>:!g++ % -g && (ulimit -c unlimited; ./a.out < ~/input.txt) <CR>
map <F5> <F6>
colo pablo
map <F12> :!gdb ./a.out -c core <CR>
```

template.cpp

```
#include<cstdio>
#include<sstream>
#include<cstdlib>
#include<cctype>
#include<cmath>
#include<algorithm>
#include<set>
#include<queue>
#include<stack>
#include<list>
#include<iostream>
#include<string>
#include<vector>
#include<cstring>
#include<map>
#include<cassert>
#include<climits>
using namespace std;

#define REP(i,n) for(int i=0, _e(n); i<_e; i++)
#define FOR(i,a,b) for(int i(a), _e(b); i<=_e; i++)
#define FORD(i,a,b) for(int i(a), _e(b); i>=_e; i--)
#define FORIT(i, m) for ( __typeof((m).begin()) i=(m).begin(); i!=(m).end(); ++i)
#define SET(t,v) memset((t), (v), sizeof(t))
#define ALL(x) x.begin(), x.end()
#define UNIQUE(c) (c).resize( unique( ALL(c) ) - (c).begin() )

#define sz size()
#define pb push_back
#define VI vector<int>
```

```
#define VS vector<string>
```

```
typedef long long LL;
```

```
typedef long double LD;
```

```
typedef pair<int,int> pii;
```

```
#define D(x) if(1) cout << __LINE__ <<" "<< #x " = " << (x) << endl;
```

```
#define D2(x,y) if(1) cout << __LINE__ <<" "<< #x " = " << (x) \
<<" , " << #y " = " << (y) << endl;
```

Combinatorics

Mathematical Sums

$$\begin{aligned} \sum_{k=0}^n k &= n(n+1)/2 & \sum_{k=a}^b k &= (a+b)(b-a+1)/2 \\ \sum_{k=0}^n k^2 &= n(n+1)(2n+1)/6 & \sum_{k=0}^n k^3 &= n^2(n+1)^2/4 \\ \sum_{k=0}^n k^4 &= (6n^5 + 15n^4 + 10n^3 - n)/30 & \sum_{k=0}^n k^5 &= (2n^6 + 6n^5 + 5n^4 - n^2)/12 \\ \sum_{k=0}^n x^k &= (x^{n+1} - 1)/(x - 1) & \sum_{k=0}^n kx^k &= (x - (n+1)x^{n+1} + nx^{n+2})/(x - 1)^2 \end{aligned}$$

Binomial coefficients

	0	1	2	3	4	5	6	7	8	9	10	11	12	
0	1													$\binom{n}{k} = \frac{n!}{(n-k)!k!}$
1	1	1												$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$
2	1	2	1											$\binom{n}{k} = \frac{n}{n-k} \binom{n-1}{k}$
3	1	3	3	1										$\binom{n}{k} = \frac{n-k+1}{k} \binom{n}{k-1}$
4	1	4	6	4	1									$\binom{n+1}{k} = \frac{n+1}{n-k+1} \binom{n}{k}$
5	1	5	10	10	5	1								$\binom{n}{k+1} = \frac{n-k}{k+1} \binom{n}{k}$
6	1	6	15	20	15	6	1							
7	1	7	21	35	35	21	7	1						
8	1	8	28	56	70	56	28	8	1					$\sum_{k=1}^n k \binom{n}{k} = n2^{n-1}$
9	1	9	36	84	126	126	84	36	9	1				$\sum_{k=1}^n k^2 \binom{n}{k} = (n+n^2)2^{n-2}$
10	1	10	45	120	210	252	210	120	45	10	1			
11	1	11	55	165	330	462	462	330	165	55	11	1		
12	1	12	66	220	495	792	924	792	495	220	66	12	1	$\binom{m+n}{r} = \sum_{k=0}^r \binom{m}{k} \binom{n}{r-k}$
	0	1	2	3	4	5	6	7	8	9	10	11	12	$\binom{n}{k} = \prod_{i=1}^k \frac{n-k+i}{i}$

Catalan numbers $C_n = \frac{1}{n+1} \binom{2n}{n}$. $C_0 = 1$, $C_n = \sum_{i=0}^{n-1} C_i C_{n-1-i}$. $C_{n+1} = C_n \frac{4n+2}{n+2}$.

$C_0, C_1, \dots = 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, \dots$

C_n is the number of: properly nested sequences of n pairs of parentheses; rooted ordered binary trees with $n+1$ leaves; triangulations of a convex $(n+2)$ -gon.

Derangements . Number of permutations of $n = 0, 1, 2, \dots$ elements without fixed points is $1, 0, 1, 2, 9, 44, 265, 1854, 14833, \dots$. Recurrence: $D_n = (n-1)(D_{n-1} + D_{n-2}) = nD_{n-1} + (-1)^n$. Corollary: number of permutations with exactly k fixed points is $\binom{n}{k} D_{n-k}$.

Stirling numbers of 1st kind . $s_{n,k}$ is $(-1)^{n-k}$ times the number of permutations of n elements with exactly k permutation cycles. $\left[\begin{smallmatrix} n \\ k \end{smallmatrix} \right] = |s_{n,k}| = |s_{n-1,k-1}| + (n-1)|s_{n-1,k}|$ $s(0,0) = 1$ and $s(n,0) = s(0,n) = 0$.

Stirling numbers of 2nd kind . $S_{n,k}$ is the number of ways to partition a set of n elements into exactly k non-empty subsets. $\left\{ \begin{smallmatrix} n \\ k \end{smallmatrix} \right\} = S_{n,k} = S_{n-1,k-1} + kS_{n-1,k}$. $S_{n,1} = S_{n,n} = 1$.

Bell numbers . B_n is the number of partitions of n elements. $B_0, \dots = 1, 1, 2, 5, 15, 52, 203, 877, \dots$ $B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_k = \sum_{k=1}^{n+1} S_{n,k}$. Bell triangle: $B_r = a_{r,1} = a_{r-1,r-1}$, $a_{r,c} = a_{r-1,c-1} + a_{r,c-1}$.

Eulerian numbers . $E(n,k) = \left\langle \begin{smallmatrix} n \\ k \end{smallmatrix} \right\rangle$ is the number of permutations with exactly k descents ($i : \pi_i < \pi_{i+1}$) / ascents ($\pi_i > \pi_{i+1}$) / excedances ($\pi_i > i$) / $k+1$ weak excedances ($\pi_i \geq i$).

Formula: $E(n, m) = (m + 1)E(n - 1, m) + (n - m)E(n - 1, m - 1)$. $E(n, 0) = E(n, n - 1) = 1$.
 $E(n, m) = \sum_{k=0}^m (-1)^k \binom{n+1}{k} (m + 1 - k)^n$.

Double factorial . Permutations of the multiset $\{1, 1, 2, 3, \dots, n, n\}$ such that for each k , all the numbers between two occurrences of k in the permutation are greater than k . $(2n - 1)!! = \prod_{k=1}^n (2k - 1)$.

Eulerian numbers of 2^{nd} kind . Related to Double factorial, number of all such permutations that have exactly m ascents. $\langle \langle \binom{n}{m} \rangle \rangle = (2n - m - 1) \langle \langle \binom{n-1}{m-1} \rangle \rangle + (m + 1) \langle \langle \binom{n-1}{m} \rangle \rangle$. $\langle \langle \binom{n}{0} \rangle \rangle = 1$

Multinomial theorem . $(a_1 + \dots + a_k)^n = \sum \binom{n}{n_1, \dots, n_k} a_1^{n_1} \dots a_k^{n_k}$, where $n_i \geq 0$ and $\sum n_i = n$.
 $\binom{n}{n_1, \dots, n_k} = M(n_1, \dots, n_k) = \frac{n!}{n_1! \dots n_k!}$. $M(a, \dots, b, c, \dots) = M(a + \dots + b, c, \dots) M(a, \dots, b)$

Data structures

BIT - Binary indexed trees

```
int bit[M], n;
void update(int x, int v) { while( x <= n ) { bit[x] += v; x += x & -x; } }
int sum(int x) { int ret=0; while(x>0){ ret += bit[x]; x -= x & -x; } return ret; }
```

BIT 2D

```
int bit[M][M], n;
int sum( int x, int y ){
    int ret = 0;
    while( x > 0 ){
        int yy = y; while( yy > 0 ) ret += bit[x][yy], yy -= yy & -yy;
        x -= (x & -x);
    }
    return ret ;
}
void update(int x , int y , int val){
    int y1;
    while (x <= n){
        y1 = y;
        while (y1 <= n){ bit[x][y1] += val; y1 += (y1 & -y1); }
        x += (x & -x);
    }
}
```

RMQ DP

```
int make_dp(int n) { // N log N
    REP(i,n) H[i][0]=i;
    for(int l=0,k; (k=1<<l) < n; l++) for(int i=0;i+k<n;i++)
        H[i][l+1] = A[H[i][l]] > A[H[i+k][l]] ? H[i+k][l] : H[i][l];
} // query log N almost O(1)
int query_dp(int a, int b) {
    for(int l=0;;l++) if(a+(1<<l+1) > b) {
        int o2 = H[b-(1<<l+1)+1][l];
        return A[H[a][l]] < A[o2] ? H[a][l] : o2;
    } }
}
```

RMQ segment tree

```

const int M = 100005;
int n, in[M], f[M], st[M], en[M];
struct data { int l, r, ans, next_l, next_r; };
data d[4*M]; // which is the range? :S
int nd;
int build( int l, int r, int id ) {
    d[ id ].l = l, d[ id ].r = r;
    if( l == r ) d[ id ].ans = f[l];
    else {
        int bar = ( r-l ) / 2 + l;
        d[id].next_l = ++nd, d[id].next_r = ++nd;
        int left = build( l, bar, d[id].next_l );
        int right = build( bar+1, r, d[id].next_r );
        d[id].ans = max( left, right );
    }
    return d[ id ].ans;
}
int query( int l, int r, int id = 0 ) {
    if( l > r ) return 0;
    if( d[id].l == l && d[id].r == r ) return d[id].ans;
    else {
        int bar = (d[id].r-d[id].l) / 2 + d[id].l;
        int left = 0, right = 0;
        if( l <= bar ) {
            if( r <= bar ) left = query( l, r, d[id].next_l );
            else {
                left = query( l, bar, d[id].next_l );
                right = query( bar+1, r, d[id].next_r );
            }
        } else right = query( l, r, d[id].next_r );
        return max( left, right );
    }
}

```

String algorithms (SuffixA, Aho-Corasick, KMP)

Suffix arrays

```

const int N = 100 * 1000 + 10;
char str[N]; bool bh[N], b2h[N];
int rank[N], pos[N], cnt[N], next[N], lcp[N];
bool smaller(int a, int b) { return str[a]<str[b];}
void suffix_array(int n) {
    REP(i,n) pos[i]=i, b2h[i]=false;
    sort(pos,pos+n,smaller);
    REP(i,n) bh[i]=!i||str[pos[i]] != str[pos[i-1]];
    for(int h=1;h<n;h*=2) {
        int buckets=0;
        for(int i=0,j; i<n; i=j) {
            j=i+1;
            while(j<n && !bh[j]) j++;
            next[i]=j;
            buckets++;
        }
    }
}

```

```

    }
    if(buckets==n)break;
    for(int i=0;i<n;i=next[i]) {
        cnt[i] = 0;
        FOR(j, i, next[i]-1) rank[pos[j]]=i;
    }
    cnt[rank[n-h]]++;
    b2h[rank[n-h]]=true;
    for(int i=0;i<n;i=next[i]) {
        FOR(j, i, next[i]-1) {
            int s = pos[j]-h;
            if(s>=0){
                rank[s] = rank[s] + cnt[rank[s]]++;
                b2h[rank[s]]=true;
            }
        }
        FOR(j, i, next[i]-1) {
            int s = pos[j]-h;
            if(s>=0 && b2h[rank[s]])
                for(int k=rank[s]+1;!bh[k] && b2h[k]; k++) b2h[k]=false;
        }
    }
    REP(i,n) pos[rank[i]]=i, bh[i]=b2h[i];
}
REP(i,n) pos[rank[i]]=i;
}
void get_lcp(int n) {
    lcp[0]=0;
    int h=0;
    REP(i,n) if(rank[i]) {
        int j=pos[rank[i]-1];
        while(i+h<n && j+h<n && str[i+h] == str[j+h]) h++;
        lcp[rank[i]]=h;
        if(h)h--;
    }
}
}

```

Aho-Corasick

```

#define NC 26
#define NP 10005
#define M 100005
#define MM 500005
char a[M];
char b[NP][105];
int nb, cnt[NP], lenb[NP], alen;
int g[MM][NC], ng, f[MM], marked[MM];
int output[MM], pre[MM];
#define init(x) {REP(_i,NC)g[x][_i] = -1; f[x]=marked[x]=0; output[x]=pre[x]=-1; }
void match() {
    ng = 0;
    init( 0 );
    // part 1 - building trie
    REP(i,nb) {
        cnt[i] = 0;
        int state = 0, j = 0;

```

```

while(g[state][b[i][j]] != -1 && j < lenb[i]) state = g[state][b[i][j]], j++;
while( j < lenb[i] ) {
    g[state][ b[i][j] ] = ++ng;
    state = ng;
    init( ng );
    ++j;
}
if( ng >= MM ) { cerr <<"i am dying"<<endl; while(1); // suicide }
output[ state ] = i;
}
// part 2 - building failure function
queue< int > q;
REP(i,NC) if( g[0][i] != -1 ) q.push( g[0][i] );
while( !q.empty() ) {
    int r = q.front(); q.pop();
    REP(i,NC) if( g[r][i] != -1 ) {
        int s = g[r][i];
        q.push( s );
        int state = f[r];
        while( g[state][i] == -1 && state ) state = f[state];
        f[s] = g[state][i] == -1 ? 0 : g[state][i];
    }
}
// final smash
int state = 0;
REP(i,alen) {
    while( g[state][a[i]] == -1 ) {
        state = f[state];
        if( !state ) break;
    }
    state = g[state][a[i]] == -1 ? 0 : g[state][a[i]];
    if( state && output[ state ] != -1 ) marked[ state ] ++;
}
// counting
REP(i,ng+1) if( i && marked[i] ) {
    int s = i;
    while( s != 0 ) cnt[ output[s] ] += marked[i], s = f[s];
}
}

```

KMP

```

int f[ len ];
f[0] = f[1] = 0;
FOR(i,2,len) {
    int j = f[i-1];
    while( true ) {
        if( s[j] == s[i-1] ) { f[i] = j + 1; break;
        }else if( !j ) { f[i] = 0; break;
        }else j = f[j];
    }
}
i = j = 0;

```

```

while( true ) {
    if( i == len ) break;
    if( text[i] == s[j] ) { i++, j++;
        if( j == slen ) // match found
    }else if( j > 0 ) j = f[j];
    else i++;
}

```

Graph algorithms (LCA, SCC, NetFlow, MinCost, BPM)

Tarjan's offline LCA

```

function TarjanOLCA(u)
    MakeSet(u); u.ancestor := u;
    for each v in u.children do
        TarjanOLCA(v); Union(u,v); Find(u).ancestor := u;
    u.colour := black;
    for each v such that {u,v} in P and v.color==black do
        print "LCA", u, v, Find(v).ancestor

```

Tarjan's Strong Connected Components

```

procedure tarjan(v)
    index = count; v.lowlink = count++; S.push(v); color[v] = 1;
    for all (v, v2) in E do
        if (!color[v2])
            tarjan(v2); v.lowlink = min(v.lowlink, v2.lowlink);
        else if (color[v2]==1)
            v.lowlink = min(v.lowlink, v2.lowlink);
    if (v.lowlink == index)
        do { v2 = S.top(); S.pop(); print v2; color[v2]=2; } while (v2 != v);
for all v in V do if(!color[v]) tarjan(v);

```

Bipartite matching with Konig

```

#define M 1010
int grid[M][M], l[M], r[M], seen[M], rows, cols;
bool dfs(int x)
{
    if( seen[x] ) return false;
    seen[x] = true;
    Rep(i,cols) if( grid[x][i] ) if( r[i] == -1 || dfs( r[i] ) )
    {
        r[i] = x, l[x] = i;
        return true;
    }
    return false;
}
int bpm() {
    SET( l, -1 );
    SET( r, -1 );
    int ret = 0;
    Rep(i,rows) {

```

```

        SET( seen, 0 );
        if( dfs( i ) ) ret ++;
    }
    return ret;
}
bool lT[M], rT[M];
void konigdfs(int x)
{
    if( !lT[x] ) return; lT[x] = 0;
    Rep(i,cols) if(grid[x][i] && i != l[x])
    {
        rT[i] = true;
        if( r[i] != -1) konigdfs(r[i]);
    }
}
int konig()
{
    SET(lT, 1); SET(rT, 0);
    Rep(i,rows) if(l[i] == -1) konigdfs(i);
}

```

Network flow - Slow

```

#define M 750
int nr, nc, o = 355, source = 740, sink = 741;
vector<int> edge[M];
int cap[M][M];
bool vis[M];
void init() {
    REP(i,M) edge[i].clear();
    SET( cap, 0 );
}
void add( int a, int b, int c, int d ) {
    edge[a].pb(b), edge[b].pb(a);
    cap[a][b] += c, cap[b][a] += d;
}
int dfs( int src, int snk, int fl ) {
    if( vis[src] ) return 0;
    if( snk == src ) return fl;
    vis[src] = 1;

    REP(i,edge[src].sz) {
        int v = edge[src][i];
        int x = min( fl, cap[src][v] );
        if( x > 0 ) {
            x = dfs( v, snk, x );
            if( !x ) continue;
            cap[src][v] -= x;
            cap[v][src] += x;
            return x;
        }
    }
    return 0;
}

```



```

}
int flow( int src, int snk ) {
    int ret = 0;
    while( 1 ) {
        SET( vis, 0 );
        int delta = dfs( src, snk, 1<<30 );
        if( !delta ) break;
        ret += delta;
    }
    return ret;
}

```

Network flow - Dinic fast

```

const int maxN = 5005;
const int maxE = 70000;
const int inf = 1000000005;
int nnode, nedge, src, snk;
int Q[ maxN ], pro[ maxN ], fin[ maxN ], dist[ maxN ];
int flow[ maxE ], cap[ maxE ], to[ maxE ], next[ maxE ];
void init( int _nnode, int _src, int _snk ) {
    nnode = _nnode, nedge = 0, src = _src, snk = _snk;
    FOR(i,1,nnode) fin[i] = -1;
}
void add( int a, int b, int c1, int c2 ) {
    to[nedge]=b, cap[nedge]=c1, flow[nedge]=0, next[nedge]=fin[a], fin[a]=nedge++;
    to[nedge]=a, cap[nedge]=c2, flow[nedge]=0, next[nedge]=fin[b], fin[b]=nedge++;
}
bool bfs() {
    SET( dist, -1 );
    dist[src] = 0;
    int st = 0, en = 0;
    Q[en++] = src;
    while( st < en ) {
        int u = Q[ st++ ];
        for(int e = fin[u]; e >= 0; e = next[e] ) {
            int v = to[e];
            if( flow[e] < cap[e] && dist[v] == -1 ) {
                dist[v] = dist[u] + 1;
                Q[en++] = v;
            }
        }
    }
    return dist[snk] != -1;
}
int dfs(int u, int fl) {
    if( u == snk ) return fl;
    for( int& e = pro[u]; e >= 0; e = next[e] ) {
        int v = to[e];
        if( flow[e] < cap[e] && dist[v] == dist[u]+1 ) {
            int x = dfs( v, min( cap[e] - flow[e] , fl ) );
            if( x > 0 ) {
                flow[ e ] += x, flow[ e^1 ] -= x;
            }
        }
    }
}

```

```

        return x;
    }
}
}
return 0;
}
LL dinic() {
    LL ret = 0;
    while( bfs() ) {
        FOR(i,1,nnode) pro[i] = fin[i];
        while( 1 ) {
            int delta = dfs( src, inf );
            if( !delta ) break;
            ret += delta;
        }
    }
    return ret;
}

```

Min-Cost Max Flow

```

#define N 705
int n, nE;
int d[N], pre[N];

struct edge {
    int to, cost, cap;
    int back;
};

edge E[N*N];
vector< int > e[N];

int mincost( int s, int t, int lim ) {

    int flow = 0, ret = 0;
    while( flow < lim ) {

        SET( d, -1 ); SET( pre, -1 );
        d[s] = 0;
        // cout <<"source "<< s <<" sink " << t << endl;
        // bellman ford
        int jump = n-1;
        bool done = 0;
        while( !done && --jump >= 0 ) {
            done = 1;
            REP(i,n) if( d[i] != -1 ) REP(j,e[i].sz) {
                edge& x = E[ e[i][j] ];
                int v = x.to;
                if( x.cap > 0 && ( d[v] == -1 || d[v] > d[i] + x.cost )) {
                    d[v] = d[i] + x.cost;
                    pre[v] = x.back;
                    done = 0;
                }
            }
        }
    }
}

```

```

//          cout<<v<<" "<<d[v]<<endl;
        }
    }
    if( done ) break;
}
//  cout << d[t] << endl;
//  if( d[t] == -1 ) break;
//  cout <<"found one path "<<endl;
//  traverse back
int x = t, cflow = 1<<30;
while( x != s ) {
    edge& ed = E[ pre[x] ];
    cflow = min( cflow, E[ ed.back ].cap );
//    cout << ed.to <<" to "<< x << endl;
    x = ed.to;
}
if( !cflow ) break;
int take = min( lim - flow, cflow );
ret += d[t] * take;
flow += take;
//  cout <<"taken flow "<< take <<" with cost "<< d[t] * take << endl << endl;
x = t;
while( x != s ) {
    edge& back = E[ pre[x] ];
    edge& forw = E[ back.back ];
    back.cap += take;
    forw.cap -= take;
    x = back.to;
}
}
//  cout << "total flow " << flow << endl;
//  if( flow < lim ) return -1;
//  return ret;
}
// remember to add -cost in the opposite direction
void add( int u, int v, int uv, int vu, int fuv, int fvu ) {
    int a = nE, b = nE+1;
    nE += 2;
    E[ a ].to = v, E[ a ].cost = uv, E[ a ].cap = fuv, E[ a ].back = b;
    E[ b ].to = u, E[ b ].cost = vu, E[ b ].cap = fvu, E[ b ].back = a;
    e[ u ].pb( a ), e[ v ].pb( b );
}

```

Geometry

Circle using three points Let $A = (0,0)$ centers are $(C_y(B_x^2 + B_y^2) - B_y(C_x^2 + C_y^2))/D'$ and $(B_x(C_x^2 + C_y^2) - C_x(B_x^2 + B_y^2))/D$ where $D = 2(B_x C_y - B_y C_x)$.

Appendices (ASCII table)

32 <small>0010 0000</small>	33 <small>0010 0001</small>	34 <small>0010 0010</small>	35 <small>0010 0011</small>	36 <small>0010 0100</small>	37 <small>0010 0101</small>	38 <small>0010 0110</small>	39 <small>0010 0111</small>	40 <small>0010 1000</small>	41 <small>0010 1001</small>	42 <small>0010 1010</small>	43 <small>0010 1011</small>	44 <small>0010 1100</small>	45 <small>0010 1101</small>	46 <small>0010 1110</small>	47 <small>0010 1111</small>
SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
48 <small>0011 0000</small>	49 <small>0011 0001</small>	50 <small>0011 0010</small>	51 <small>0011 0011</small>	52 <small>0011 0100</small>	53 <small>0011 0101</small>	54 <small>0011 0110</small>	55 <small>0011 0111</small>	56 <small>0011 1000</small>	57 <small>0011 1001</small>	58 <small>0011 1010</small>	59 <small>0011 1011</small>	60 <small>0011 1100</small>	61 <small>0011 1101</small>	62 <small>0011 1110</small>	63 <small>0011 1111</small>
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64 <small>0100 0000</small>	65 <small>0100 0001</small>	66 <small>0100 0010</small>	67 <small>0100 0011</small>	68 <small>0100 0100</small>	69 <small>0100 0101</small>	70 <small>0100 0110</small>	71 <small>0100 0111</small>	72 <small>0100 1000</small>	73 <small>0100 1001</small>	74 <small>0100 1010</small>	75 <small>0100 1011</small>	76 <small>0100 1100</small>	77 <small>0100 1101</small>	78 <small>0100 1110</small>	79 <small>0100 1111</small>
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80 <small>0101 0000</small>	81 <small>0101 0001</small>	82 <small>0101 0010</small>	83 <small>0101 0011</small>	84 <small>0101 0100</small>	85 <small>0101 0101</small>	86 <small>0101 0110</small>	87 <small>0101 0111</small>	88 <small>0101 1000</small>	89 <small>0101 1001</small>	90 <small>0101 1010</small>	91 <small>0101 1011</small>	92 <small>0101 1100</small>	93 <small>0101 1101</small>	94 <small>0101 1110</small>	95 <small>0101 1111</small>
P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
96 <small>0110 0000</small>	97 <small>0110 0001</small>	98 <small>0110 0010</small>	99 <small>0110 0011</small>	100 <small>0110 0100</small>	101 <small>0110 0101</small>	102 <small>0110 0110</small>	103 <small>0110 0111</small>	104 <small>0110 1000</small>	105 <small>0110 1001</small>	106 <small>0110 1010</small>	107 <small>0110 1011</small>	108 <small>0110 1100</small>	109 <small>0110 1101</small>	110 <small>0110 1110</small>	111 <small>0110 1111</small>
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112 <small>0111 0000</small>	113 <small>0111 0001</small>	114 <small>0111 0010</small>	115 <small>0111 0011</small>	116 <small>0111 0100</small>	117 <small>0111 0101</small>	118 <small>0111 0110</small>	119 <small>0111 0111</small>	120 <small>0111 1000</small>	121 <small>0111 1001</small>	122 <small>0111 1010</small>	123 <small>0111 1011</small>	124 <small>0111 1100</small>	125 <small>0111 1101</small>	126 <small>0111 1110</small>	127 <small>0111 1111</small>
p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL