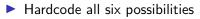
TITLEPAGE

Bc., Ing., Ph.D.

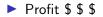
October 22, 2018

Die



Rotate





```
int x;
int began;
forfint intistist...) f
     beg+tx;
if(i<i)
         beport#;
stroyl.insert((bep. 1.0));
vector-double= prs(2);
ist plup: // p white, 1 black
while(true) {
     for(auto ## s: stavy1) (
        // if(s.second==0) continue; // WTF
        int steves.first;
         int x, y, x2, y2;
        (f(pl=0) {
             (pc110) {
x1(stex>24)$255;
y1(stex>26)$255;
        yr(ttavjarss,
)else[
x2*(stmv=24)&255;
y2*(stmv=16)&255;
x*(stmv=6)&255;
       lat no newsrit:
         set-pair(int,int>> taby = { (-2, -1),
         for(auto 86 t: tahy) (
              if(non1 55 need 55 npon) 55 npont)
                 no newsrr;
        for(auto && t: tahy) (
             nyry+t.second;
if(nxx+1 && nxx+8 && nyx+1 && nyx+8) (
                int newState=0;
if(p]==0) (
                      newStater+nx++24;
                    new(Late=Decold)
                      newState+=y2;
                  ]else(
                      newitate+m2<<24:
                       resitaterenced;
                      newStater-my;
(/contennes' 'enres' 'enrettates'AAAA'earafi)
                 if(narms2 && nyrmy2) {
    //costec*PLUS*ex.second/no_newsee* "explorendl;
    prs[sl]+ms.second/no_news;
                  letze
                  stavy2[newState]+=s.second/no news;
    // sun
double sunce.e;
     for(auto ## s: stavy2) (
     if(sum=0.000001) break;
     plan-pl;
     stavinstd::mye(stav2);
//costeeprs(0)ee' 'eeprs[1];
if(prs[0]-prs[1]) cost-<"black"-cendl;
return 6:
```

int s;	× ×
int begang Forfint Linelageiret f	7 Adeline p0 push back s Zdefine ng nake pair
CANNAL DEGATAL	2
(ff(a))	10 typedef vector <int> vi; 11 typedef pair<int,int> pli;</int,int></int>
begectii;	12 tagetef loss loss 11:
strevi insert(Den 1.0)):	13 typedef long double ld;
ectorvdoble= prs(2);	15 const int mean = 3000 + 5; 10
ectorodoublex prs(2); int pl=0; // 0 white, 1 black	17 typedef array-array-ld, 8+, 8+ stan; 18
dile(true) {	10 - stan pusty_stan() [10 - stan s]
// nove for(auto && s: stavy1) (21 rep (1, 0] rep(1, 0] a[1][1] = 0.0;
// if(s.second==0) continue; // WTF	22 return s; 23 3
int steves.first;	24 35 - bool in rempetint a) C
int x,y, x2, y2;	20 return x >=0 65 x < 0;
(f(plmm) { x=(stmm>24)8235;	27 } 28
y=(stex>>0)\$235;	<pre>29 + bool welld(int i, int j, int de, int dy) { 10 bool keen ((abcdds) = 2 &k abc(dy) = 1) (abcdds) == 1 &k abc(dy) == 2)]; 11 return hom is to,range(dw + 1) &k (n_range(dw + j)); 12 return hom is to,range(dw + 1) &k (n_range(dw + j)); 13 return hom is to,range(dw + 1) &k (n_range(dw + j)); 14 return hom is to,range(dw + 1) &k (n_range(dw + j)); 15 return hom is to,range(dw + 1) &k (n_range(dw + j)); 16 return hom is to,range(dw + 1) &k (n_range(dw + j)); 17 return hom is to,range(dw + 1) &k (n_range(dw + j)); 18 return hom is to,range(dw + 1) &k (n_range(dw + j)); 19 return hom is to,range(dw + 1) &k (n_range(dw + j)); 10 return hom is to,range(dw + 1) &k (n_range(dw + j)); 11 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 12 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 13 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 14 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 15 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 16 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 17 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 18 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1) &k (n_range(dw + 1)); 19 return hom is to,range(dw + 1)); 19 return hom is to,range(dw +</pre>
x2r(stavo-4)4255; x2r(stav)4255;	<pre>10 Dool Kon = ((acc(dx) = 2 & acc(dy) = 1)) (acc(dx) = 1 & acc(dy) = 2)); 11 return kon & in_range(dx + 1) & in_range(dy + j);</pre>
Jelsel	12 3
x2=(stmv=24)A355; x2=(stmv=16)A355;	34 v stan kolginy(const stank obecny, const stank drupt) (
x=(stm==0)8255; y=(stm=)8255;	<pre>35 stan res = pusty_stan(); 30</pre>
y=(ster)8235;	37 - res(i, 8) (38 - res(i, 8) (
lat as newsrif:	
set-pair <int,int>> taby = { (-2, -1),</int,int>	$\begin{array}{l} a_{1} & \\ a_{2} & \\ a_{3} & \\ a_{4} & \\$
(-2, +1),	42 43 of (ivelid(i, j, k, l)) continue; 43 pos.push back((i = k, j = U));
(+2, -1), (+2, +1),	
(-1, -1),	45 3
(-1, +2), (+1, -2),	47 Id ile = (10)pos.size(); 48 - for (auto p : pos) [
(+1, +2) };	
for(auto && t: taby) (<pre>int y = p.second; in</pre>
int nx_ny; nxxx+i.first;	
nyvy+t.second; if(nxx+1 && nxx+8 && nyx+1 && nyx+8)	54
	55 56 return res;
)	17 }
for(auto && t: taby) { int nx_ny;	15 15 - paireld, ld- ps(int iter, stans pracas, stans pracas, ld wyniks, ld wyniks) [
axtest.first:	00 //orinff('s: NLF, b: NLF(n', synikk, synikk); 01 - If (iter >= 60) {
nyryst.second; if(nost1 AA nost8 AA nost1 AA nost8) (cl return (wynikk, wynikk);
int newState=0;	63) 64
if(pl==0) (newState==new=24;	<pre>05 outo nowpA = kolejny(graczA, graczB); 00 - rep(i, B) (</pre>
rewState=Imp<<00 rewState=Im2<<8)	
new1Late+Ty2;	<pre>co wgwiAA == nowyA[i][i] * grecz0[i][i]]; co</pre>
<pre>}else(newitate+rs2<<26;</pre>	70 3
newitabe=ty2<<10; newitabe=ty2<<10;	72 auto noveli = koleinv(graczł, noveł);
newState==my;	73 = rep(1, 8) (74 = rep(3, 8) (
	<pre>75 septids == noop4(i](j] = noop4(i](j); 76</pre>
Houtenee' 'emper' 'empdiatec'AAAA'eendij	77
<pre>if(narms2 && nyrmy2) { //coutes"Pull"ext.second/no_newtes" "explorendl;</pre>	<pre>TR TP return go[iter + 1, noup1, noup8, wynik1, wynik8);</pre>
prs[pl]+ts_second/no_news;	80 3
<pre>}else stavy2[newState]+=s.second/no_news;</pre>	star ist nate(M
)	11 tot a, b; 14 tot p, q;
	55 scorf ('Nd Nd Nd Nd', 6a, 8b, 8a, 6a);
1	
// sun double sunce.e:	18. Inc. 19. Inc.
for(auto ## s: stavy2) (10 P - 1 10 Q - 1 11
survers.second;	92 auto graciA = pusty_stan();
·	<pre>93 graczA[a][b] = 1.0; 94</pre>
if(sur-0.000001) break;	
plus-pl;	90 proced(s)[s] = 1.0; 97
	<pre>90 euto wymik = ge(0, precz4, precz8, 0.0, 0.0); 99</pre>
stavyinstd::move(stavy2); stavy2.clear();	<pre>//printf('mif, mif(n', synik.first, synik.second); 101</pre>
	102 - if (abs(wymik.first - wymik.second) < (id)0.000001) (
//costeeprs[0]ee' 'eeprs[1];	
((and [b] and [1]) contro [b] ack [contro]	<pre>104 +) else if (wynik.first > wynik.second) (105</pre>
	<pre>bit = } else if (weak.first > wysk.second) { bit = post("wist"; bit = post("wist"; post("black"); post("black"); bit = po</pre>

int x; int begins		
beg+tx; lf(L<8)		
bepectik;		
)		
stavyl.insert((beg, 1.0));		
vector-double= prs(2); ist pl=0; // 0 white, 1 black		
while(true) { // nove		
for(auto && s: stavy1) (// tf(s.second==0) conf		
// tf(s.second==0) cont	tinue; // WTF	
int steves first;		
int x,y, x2, y2;		
(f(plm0) { x1(sterrozd)\$255;		
x1(61#4>>24)6255) y1(61#4>>30)6255)		
x2+(stav>+8)8255;		
)else[
x2+(stmv+24)8255;		
y2+(stmv+16)&255; x+(stmv+0)&255;		
y=(stev)8255;		
int no_newsmip		
set-pair-int, int>> tab;		
set-pair-(int, int>> tab;	(= ((-2, -1), (-2, +1),	
	(+1, +2), (+1, -2),	
	(+1, +2) };	
for(auto && t: tahy) { int nx,ny;		
nxvs+t.first;		
if(no=1 & no=5 &	A ny++1 AA ny++8)	
no_news++;		
for(auto && t: tahy) {		
nxvs+t.first;		
if(mort) A& morth #	A mont AA month) (
if(pl==0) (
nextater-r	uev24;	
newState+2	(2)	
]else(
newitate+m newitate+m	2+(24)	
newState+n	most:	
	95	
)		
	'onyor' 'constituteor'AMM'condi;	
if (name2 44 pr	rmy2) { <s.second "explorentl;<br="" no_news<*="">second/no_news;</s.second>	
//cout <<"PLUS"	<pre>cs.second/no_news<<" "explorendl;</pre>	
prs[pl]+ns. Jelse	second/no_news;	
staw21 april ats	l++s.second/no news;	
)	,	
1		
// sun double sunce.e;		
1		
if(sum=0.000001) break;		
plai-plj		
stavyinstd::nove(stavy2);		
)		
//contemps[0]es' 'empt[1];		
if(prs[0]eprs[1]) costsc/black*coend(; else costsc/white*coend(;		
else coutoc'ubite'ocendi;		
return 0;		

7 | #defute p0 push back typedef vector-clata via const int mean = 3000 + 5; 17 typedef arres-arres-ld, 8+, 8+ star; 17 * stan pusty stan() (rep (i, #) rep(j, #) s[i][j] = 0.0; return s; 25 - bool in range(int x) (return x >=0 66 x + 5; 29 - bool walid(int i, int 5, int dw, int dy) { bolk kon = ((sbu(dx) == 2 && sbu(dy) == 1) || (sbu(dx) == 1 && sbu(dy) == 2));
return kon && to, range(dx + 1) && to, range(dy + 1); 14 - stan koleinv(const stank obecny, const stank drugi) (aten rea = pusty aten[]; rep(i, 8) (rep(i, 8) (P(2, 0) t vector-pll> pog; for {lat k = -2; k <= 2; k++) { for (lat l = -2; l <= 2; l++) { if (velid(t, j, k, l)) continue; pog.publbeck((t + k, j + l)); ld ile = (ld)aus_size(): for (auto p : pos) { res[x][y] += obecne[i][i] + ((id)1.0 - obecne[i][i] + drugi[i][i]) / ile: 17 + paireld, ld- po(int iter, stant gracal, stant gracal, ld wowike, ld wowike) { //orintf('s: %Lf, b: %Lf(n', wynikk, wynikk); if (iter >: ob) { return Gentiks, works); suto nowyA = kolejny(graczA, graczB); rep(j, 8) ((j, 8) {
 wymikk == nowyk[i][j] * gracz0[i][j]; auto novy8 = kalejny(gracz8, novy8); rep(i, i) (rep(j, i) (ive.1, 4) {
 symikt == nowpk[i][j] = nowpt[i][j];
} return go(iter + 1, nowsA, nowsB, wymikB, wymikB); sconf("Nd Nd Nd Nd", do, db, do, da); auto graczA = pusty_stan(); proceAlelibl = 1.4: euto greczil - pusty sterill: precedipile) = 1.4; esto wymik = go(0, proco4, proco5, 0.0, 0.0); if (sbs/wmik.first - wmik.second) < (1d)0.000001) (} else if (work,first > work,second) (puts('black');

5 Set n = 20000 return a+8 && a>=0; unt p = 0; if(isin(x+1) && isin(y+2)) p++; return o: vector-vector-long double> > b(8, vector-vector-long double>(8, vector-long double> (n,0))) vector-vector-vector-long double> > b(8, vector-vector-long double>(8, vector-long double> (n,0))) int bx, by, ww, wy; bk >> bk >> by >> wk >> bk >> bk >> by >> wk >> b(bx)[by[[0]=1; for(ist i = 1; ist; i++) { for(ist i = 0; ist; i++) { tf(tcin(j+1) ## tcin(k+2)) { w[j][k][t] += w[j+1][k+2][t-1]/posnoves(j+1,k+2); tf(icin(j+1) ## icin(k+2)) { b[j][k][i] += b[j+1][k+2][i-1]/posnovec(j+1,k+2); long double winny, beinny: for(int k = 0; kv0; kv+) {
 for(int k = 1; kv0; kv+) {
 for(int i = 1; kv0; iv+) {
 buine= w[j][k][i] * b[j][k][i];
 wufan= w[j][k][i] * b[j][k][i-1];
 wufan= u[j][k][i] * b[j][k][i-1];
 wufan= u[j][k][i] * b[j][k][i];
 transfer = u[j][k][i] * b[j][k][i] * b[j][k][i] * b[j][k][i];
 transfer = u[j][k][i] * b[j][k][i] * b[j] \f[fabs(wdo-bwin)-0.000001] { cout << 'draw's elar if(min a hein) (cout << "white"; else { cost << "black"; cout << endl:

Horsemeat - alternative solution

Horsemeat - alternative solution



Horsemeat - alternative solution

```
k = [2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2
           i1. i1 = [int(x)-1 for x in input().split()]
           i2, j2 = [int(x)-1 for x in input().split()]
           res = l[(i1+j1*8)*64+(i2+j2*8)]
9 * if res == 0:
                   print('draw')
11 - elif res == 1:
                   print("white")
13 · else:
                   print('black')
                                                                                  l = [2, 1, 2, 1, 2, 1, 2, 1,
                                                                                                        1. 2. 1. 2. 1. 2. 1. 2.
                                                                                                        2. 1. 2. 1. 2. 1. 2. 1.
                                                                                                        1, 2, 1, 2, 1, 2, 1, 2,
                                                                                                        2, 1, 2, 1, 2, 1, 2, 1,
                                                                                                       1, 2, 1, 2, 1, 2, 1, 2,
                                                                                                        2. 1. 2. 1. 2. 1. 2. 1.
                                                                                                       1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 2, 1, 2,
                                                               9
                                                             10
                                                                                 i1, j1 = [int(x)-1 for x in input().split()]
                                                                                  i2, j2 = [int(x)-1 for x in input().split()]
                                                                                  res = l[(i1+i1*8)*64+(i2+i2*8)]
                                                             14
                                                            16 - if res == 0:
                                                                                                  print("draw")
                                                                                 elif res == 1:
                                                             18 *
                                                                                                  print("white")
                                                             28 *
                                                                                 elser
                                                                                                  print("black")
```

Horsemeat

Observation

Horsemeat

٠

Horsemeat

6

Security Guards

Security Guards

- Fill the 5000 × 5000 grid with the distance to the nearest guard.
- Use BFS with multiple starts.
- Query in constant time

Security Guards

- Fill the 5000 × 5000 grid with the distance to the nearest guard.
- Use BFS with multiple starts.
- Query in constant time

Complexity $O(N^2 + Q)$

Security Guards - alternative solution

- Fill the 5000 × 5000 grid with 1 on guard positions and 0 otherwise.
- Create 2D prefix sum.
- For each query use binary search by answer.

Security Guards - alternative solution

- Fill the 5000 × 5000 grid with 1 on guard positions and 0 otherwise.
- Create 2D prefix sum.
- For each query use binary search by answer.

Complexity $O(N^2 + Q \log N)$

Lightning

Lightning

- Dynamic Programming
- Parameters:
 - position in bit string
 - number of ones so far
 - carry
- Expand each state by adding 1 or 0

Lightning

- Dynamic Programming
- Parameters:
 - position in bit string
 - number of ones so far
 - carry
- Expand each state by adding 1 or 0

Complexity O(NK)

- Each impartial game is equivalent to NIM.
- Sum of games has nimber equivalent to XOR of nimbers of individual games.
- Note that adding two same piles do not change the outcome.
- ▶ We precompute nimbers up to piles of size 2000 (DP).
- > XOR nimbers of games on the input.
- ▶ If XOR is zero second player wins, otherwise first player wins.

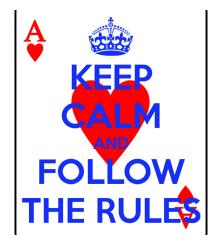
- Each impartial game is equivalent to NIM.
- Sum of games has nimber equivalent to XOR of nimbers of individual games.
- Note that adding two same piles do not change the outcome.
- ▶ We precompute nimbers up to piles of size 2000 (DP).
- > XOR nimbers of games on the input.
- ▶ If XOR is zero second player wins, otherwise first player wins.

Complexity $O(N^2 \log N)$

- Each impartial game is equivalent to NIM.
- Sum of games has nimber equivalent to XOR of nimbers of individual games.
- Note that adding two same piles do not change the outcome.
- ▶ We precompute nimbers up to piles of size 2000 (DP).
- > XOR nimbers of games on the input.
- ▶ If XOR is zero second player wins, otherwise first player wins.

Complexity $O(N^2 \log N)$

If you try hard, you can achieve $O(N^2)$



- 1. If you have at least 4 cards in your hand then add 1 to the value. Also add to the value the multiple of the number of J's in your hand and the score of the first card in your hand.
- 2. If you have at least 2 cards of the same suit in your hand then multiply the value by 2.
- 3. If you have at least one card of each suit in your hand then multiply the value by 2.
- 4. If the count of black (Clubs and Spades) and the count of red (Hearts and Diamonds) cards in your hand differ then add the absolute difference of the counts to the value.
- 5. If the value is currently even then add all positive integer divisors of the value (including 1 and the value itself) to the value.
- 6. If there are exactly 4 cards of rank 7 in your hand then subtract 11^2 from the value.
- 7. If the value is currently non-negative then add the score of the lowest score card in your hand to the value.
- 8. If the value is currently negative then multiply the value by -1.
- 9. If there are at least 3 cards of Diamond suit in your hand then add 1 to the value and swap ranks of all 6's to 9's, all 9's to 6's, all 2's to 5's, and all 5's to 2's in your hand at once.
- 10. If there is a straight in your hand then add five times the number of A's in your hand to the value.
- 11. If the value was modified by the rules more than 8 times so far then add the number of 1's bits in the binary representation of the value to the value.
- 12. If there is at least one card of rank 2 in your hand then apply once again the last rule which changed the score (after that continue with rule ??).
- 13. If there is at least one card of rank 2 in your hand then add the product of all distinct superfactors of the value to the value. A superfactor divides the value evenly and it is the highest power of a prime factor of the value.
- 14. If the value is 674 you win!

- 1. If you have at least 4 cards in your hand then add 1 to the value. Also add to the value the multiple of the number of J's in your hand and the score of the first card in your hand.
- 2. If you have at least 2 cards of the same suit in your hand then multiply the value by 2.
- 3. If you have at least one card of each suit in your hand then multiply the value by 2.
- 4. If the count of black (Clubs and Spades) and the count of red (Hearts and Diamonds) cards in your hand differ then add the absolute difference of the counts to the value.
- 5. If the value is currently even then add all positive integer divisors of the value (including 1 and the value itself) to the value.
- 6. If there are exactly 4 cards of rank 7 in your hand then subtract 11^2 from the value.
- 7. If the value is currently non-negative then add the score of the lowest score card in your hand to the value.
- 8. If the value is currently negative then multiply the value by -1.
- 9. If there are at least 3 cards of Diamond suit in your hand then add 1 to the value and swap ranks of all 6's to 9's, all 9's to 6's, all 2's to 5's, and all 5's to 2's in your hand at once.
- 10. If there is a straight in your hand then add five times the number of A's in your hand to the value.
- 11. If the value was modified by the rules more than 8 times so far then add the number of 1's bits in the binary representation of the value to the value.
- 12. If there is at least one card of rank 2 in your hand then apply once again the last rule which changed the score (after that continue with rule ??).
- 13. If there is at least one card of rank 2 in your hand then add the product of all distinct superfactors of the value to the value. A superfactor divides the value evenly and it is the highest power of a prime factor of the value.
- 14. If the value is 674 you win!

- 1. Add 1 to the value. Also add to the value the multiple of the number of J's in your hand and the score of the first card in your hand.
- 2. Multiply the value by 2.
- 3. If you have at least one card of each suit in your hand then multiply the value by 2.
- 4. Add the absolute difference between red and black suites to the value.
- 5. If there are exactly 4 cards of rank 7 in your hand then subtract 11^2 from the value.
- 6. Add the score of the lowest score card in your hand to the value.
- 7. If there are at least 3 cards of Diamond suit in your hand then add 1 to the value and swap ranks of all 6's to 9's, all 9's to 6's, all 2's to 5's, and all 5's to 2's in your hand at once.
- 8. If there is a straight in your hand then add five times the number of A's in your hand to the value.
- 9. If the value was modified by the rules more than 8 times so far then add the number of 1's bits in the binary representation of the value to the value.
- 10. If there is at least one card of rank 2 in your hand then apply once again the last rule which changed the score (after that continue with rule ??).
- 11. If there is at least one card of rank 2 in your hand then add the product of all distinct superfactors of the value to the value. A superfactor divides the value evenly and it is the highest power of a prime factor of the value.

- 1. Add 1 to the value. Also add to the value the multiple of the number of J's in your hand and the score of the first card in your hand.
- 2. Multiply the value by 2.
- 3. A: If you have at least one card of each suit in your hand then multiply the value by 2.
- 4. Add the absolute difference between red and black suites to the value.
- 5. B: If there are exactly 4 cards of rank 7 in your hand then subtract 11^2 from the value.
- 6. Add the score of the lowest score card in your hand to the value.
- 7. ¬A: If there are at least 3 cards of Diamond suit in your hand then add 1 to the value and swap ranks of all 6's to 9's, all 9's to 6's, all 2's to 5's, and all 5's to 2's in your hand at once.
- 8. $\neg B$: If there is a straight in your hand then add five times the number of A's in your hand to the value.
- 9. If the value was modified by the rules more than 8 times so far then add the number of 1's bits in the binary representation of the value to the value.
- 10. If there is at least one card of rank 2 in your hand then apply once again the last rule which changed the score (after that continue with rule ??).
- 11. If there is at least one card of rank 2 in your hand then add the product of all distinct superfactors of the value to the value. A superfactor divides the value evenly and it is the highest power of a prime factor of the value.

- 1. Add 1 to the value. Also add to the value the multiple of the number of J's in your hand and the score of the first card in your hand.
- 2. Multiply the value by 2.
- 3. If you have at least one card of each suit in your hand then multiply the value by 2.
- 4. Add the absolute difference between red and black suites to the value.
- 5. If there are exactly 4 cards of rank 7 in your hand then subtract $11^2 \mbox{ from the value.}$
- 6. Add the score of the lowest score card in your hand to the value.
- 7. If there are at least 3 cards of Diamond suit in your hand then add 1 to the value and swap ranks of all 6's to 9's, all 9's to 6's, all 2's to 5's, and all 5's to 2's in your hand at once.
- 8. If there is a straight in your hand then add five times the number of A's in your hand to the value.
- 9. If there is at least one card of rank 2 in your hand then apply once again the last rule which changed the score (after that continue with rule ??).
- 10. If there is at least one card of rank 2 in your hand then add the product of all distinct superfactors of the value to the value. A superfactor divides the value evenly and it is the highest power of a prime factor of the value.

- 1. Add 1 to the value. Also add to the value the multiple of the number of J's in your hand and the score of the first card in your hand.
- 2. Multiply the value by 2.
- 3. If you have at least one card of each suit in your hand then multiply the value by 2.
- 4. Add the absolute difference between red and black suites to the value.
- 5. If there are exactly 4 cards of rank 7 in your hand then subtract 121 from the value.
- 6. Add the score of the lowest score card in your hand to the value.
- 7. If there are at least 3 cards of Diamond suit in your hand then add 1 to the value and swap ranks of all 6's to 9's, all 9's to 6's, all 2's to 5's, and all 5's to 2's in your hand at once.
- 8. If there is a straight ending in A, add five to the value.
- 9. If there is at least one card of rank 2 in your hand then apply once again the last rule which changed the score (after that continue with rule ??).
- 10. If there is at least one card of rank 2 in your hand then multiply value by 2.

- 1. Add 1 to the value. Also add to the value the multiple of the number of J's in your hand and the score of the first card in your hand.
- 2. Multiply the value by 2.
- 3. If you have at least one card of each suit in your hand then multiply the value by 2.
- 4. Add the absolute difference between red and black suites to the value.
- 5. If there are exactly 4 cards of rank 7 in your hand then subtract 121 from the value.
- 6. Add the score of the lowest score card in your hand to the value.
- 7. If there are at least 3 cards of Diamond suit in your hand then add 1 to the value and swap ranks of all 6's to 9's, all 9's to 6's, all 2's to 5's, and all 5's to 2's in your hand at once.
- 8. If there is a straight ending in A, add five to the value.
- 9. If there is at least one card of rank 2 in your hand then apply once again the last rule which changed the score (after that continue with rule ??).
- 10. If there is at least one card of rank 2 in your hand then multiply value by 2.

Simple implementation

- 1. Add 1 to the value. Also add to the value the multiple of the number of J's in your hand and the score of the first card in your hand.
- 2. Multiply the value by 2.
- 3. If you have at least one card of each suit in your hand then multiply the value by 2.
- 4. Add the absolute difference between red and black suites to the value.
- 5. If there are exactly 4 cards of rank 7 in your hand then subtract 121 from the value.
- 6. Add the score of the lowest score card in your hand to the value.
- 7. If there are at least 3 cards of Diamond suit in your hand then add 1 to the value and swap ranks of all 6's to 9's, all 9's to 6's, all 2's to 5's, and all 5's to 2's in your hand at once.
- 8. If there is a straight ending in A, add five to the value.
- 9. If there is at least one card of rank 2 in your hand then apply once again the last rule which changed the score (after that continue with rule ??).
- 10. If there is at least one card of rank 2 in your hand then multiply value by 2.

Simple implementation ... with high probability of a mistake

Escalators

Escalators

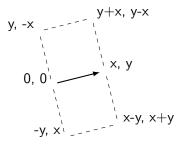
- Split the problem into 20 instances by bits.
- In i-th iteration consider only nodes which have 1 on the i-th bit.
- Find sizes of components of ones (DFS).
- Each component adds $\binom{size}{2} \cdot 2^i$.

Escalators

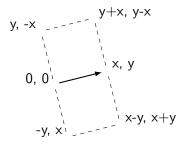
- Split the problem into 20 instances by bits.
- In i-th iteration consider only nodes which have 1 on the i-th bit.
- Find sizes of components of ones (DFS).
- Each component adds $\binom{size}{2} \cdot 2^i$.

Complexity $O(N \log W)$

Basic operation: complete square from 2 given points



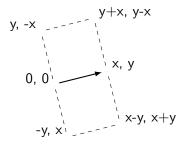
Basic operation: complete square from 2 given points





- Find lowest leftmost point
- Try to create square with each of the remaining points
- Choose the smallest square
- Erase points of the chosen square

Basic operation: complete square from 2 given points





- Find lowest leftmost point
- Try to create square with each of the remaining points
- Choose the smallest square
- Erase points of the chosen square

Complexity $O(N^2 \log N)$

• Create Suffix array over s + s.

• Create Suffix array over s + s.

- Binary search by answer
 - Scratch out every segment with lesser rank.
 - Scratching can be done with O(N) sweep.
 - Total complexity $O(N \log N)$.

• Create Suffix array over s + s.

- Binary search by answer
 - Scratch out every segment with lesser rank.
 - Scratching can be done with O(N) sweep.
 - Total complexity $O(N \log N)$.
- Keep track of non-scratched segments
 - Traverse SA and remove scratched segments.
 - Segments in Set: O(log N)
 - Total complexity $O(N \log N)$.

• Create Suffix array over s + s.

Binary search by answer

- Scratch out every segment with lesser rank.
- Scratching can be done with O(N) sweep.

• Total complexity $O(N \log N)$.

- Keep track of non-scratched segments
 - Traverse SA and remove scratched segments.
 - Segments in Set: O(log N)
 - Total complexity $O(N \log N)$.
- Note that complexity depends on SA construction.
 O(N log² N) suffices (if implemented reasonably).

• Keep track of current state with Finite automaton.

Keep track of current state with Finite automaton.

We can use Aho-Corasick algorithm to achieve this.

- Keep track of current state with Finite automaton.
- We can use Aho-Corasick algorithm to achieve this.
- We create matrix A with probabilities of transitions.
- We can achieve this by giving each edge in Aho-Corasick ¹/₂ probability of traversal, and creating edges with probability 1 from all final states to one common final state.

- Keep track of current state with Finite automaton.
- We can use Aho-Corasick algorithm to achieve this.
- ▶ We create matrix A with probabilities of transitions.
- We can achieve this by giving each edge in Aho-Corasick ¹/₂ probability of traversal, and creating edges with probability 1 from all final states to one common final state.
- Now find the expected number of steps before ending in absorbing state.

- Keep track of current state with Finite automaton.
- We can use Aho-Corasick algorithm to achieve this.
- ▶ We create matrix A with probabilities of transitions.
- We can achieve this by giving each edge in Aho-Corasick ¹/₂ probability of traversal, and creating edges with probability 1 from all final states to one common final state.
- Now find the expected number of steps before ending in absorbing state.
- After solving (E A)x = 1 the answer can be found in x_0 .

- Keep track of current state with Finite automaton.
- We can use Aho-Corasick algorithm to achieve this.
- ▶ We create matrix A with probabilities of transitions.
- We can achieve this by giving each edge in Aho-Corasick ¹/₂ probability of traversal, and creating edges with probability 1 from all final states to one common final state.
- Now find the expected number of steps before ending in absorbing state.
- After solving (E A)x = 1 the answer can be found in x_0 .
- ► To solve this equation we can use Gaussian Elimination.

- Keep track of current state with Finite automaton.
- We can use Aho-Corasick algorithm to achieve this.
- ▶ We create matrix A with probabilities of transitions.
- We can achieve this by giving each edge in Aho-Corasick ¹/₂ probability of traversal, and creating edges with probability 1 from all final states to one common final state.
- Now find the expected number of steps before ending in absorbing state.
- After solving (E A)x = 1 the answer can be found in x_0 .
- ► To solve this equation we can use Gaussian Elimination.
- ► Complexity $O((W B)^3)$

							#
							22

				*		##	**

****			***** **				
******			*******				
******			******				
##	******	**	***	raaaa	# #	#	
#	******	***	***	rxxxx	#		

XX XXXXXXXXXXXXXXXX							

XXXXXXXXXXXXXXXXX XXXX							
	****	****	****	***		**	
	*****	****	***	##		***	
	******	****	***	##	#	##	
##	******	****	***	##	#		
##	****	****	**				
#	***	#	##				
#	##	#	#				
	##	#	#				
	***	#	##				

Thank you for your attention!