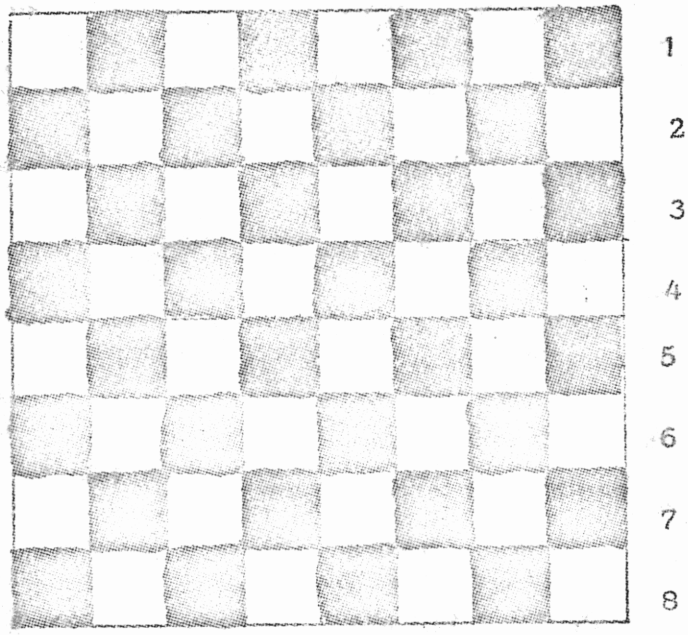
I. 2

DAMSPIL.

HVID



SORT

FRED MOSEKJÆR MADSEN,

OLE SCHELDE JACOBSEN.

Damprogrammet er programmeret i gier-algol 4, og udnytter flere af gier algols specielle faciliteter(shift og typeundertrykkelser), men det er tanken senere at flytte det over på CDC 6400. Problemerne derved bliver ikke store, lidt øget lagerkrav og et par compass procedurer er formentlig nok. Programmet spiller endnu ikke tilfredsstillende, især kniber det i slutspillet. Grunden hertil må nok søges i pointsystemet og lookahead termination, som begge et ret primitive. Løsningen af dette problem kræver nok mere evner som damspiller end som programmør, men der vil blive arbejdet med problemet.

I.2.1 (Games)
I.2.8

CR 3.60

DAMSPILLENDE PROGRAM

strategi.
dam er et endeligt spil, og der kan derfor opstilles en vindende strategi for spillet. teoretisk skulle vi således være istand til at opstille det træ hvor i alle mulige spil findes. i praksis er dette imidlertid umuligt. (Samuelson antager st det vil tage 10^{23} år at opbygge dette træ). Vi må altså gå andre veje. for os at se er der nu to principielt forskellige metoder.

1. man kan opbevare stillinger, som erfaringsmæssigt giver gode resultater, eventuelt kan man lade programmet lære af sine egne og sine modstanderes fejl. denne metode har den ulempe, at programmet ikke kan blive bedre end sine bedste modspillere. desuden vil man også hurtigt få et hav af gode stillinger at gemme på. Samuelson har således over 50.000 gode stillinger at vælge imellem.
   For os at se synes det derfor at være en bedre metode at:
2. opbygge grene af spilletræet og udfra et pointsystem af træk, stag og brikker, at bestemme det optimale træk. svagheden ved denne metode er pointsystemet, idet dette jo må bestemmes udfra erfaringen , men vi har den mulighed at kunne gøre grenene meget lange, og jo længere grenene bliver, jo mindre betydning får pointsystemet. et program opbygget efter system nr. 2 er således i princippet i stand til at vinde ethvert spil.

Som det måske fremgår af ovenstående, er den metode vi synes best om, og derfor vil benytte, metode nr. 2. på nuværende tidspunkt (primo marts) har vi ikke tænkt os at lad programmet selv modificere pointsystemet, hvis det skulle vise sig at være nødvendigt.

repræsentation af dambrættet.
brætte repræsenteres i et array b[1:32, 1:6], hvor hver række så repræsenterer et felt.
celle 3, 4, 5 og 6 i hver række angiver adresserne på nabofelterne.
celle 2 indeholder et pointtal, og celle 1 angiver i de tre førstebit hvad der står på feltet.

oplysningerne i b[i, 1] har følgende udseende:

bit \ indhold	false	true
0	fri	opt.
1	sort	hvid
2	alm.	dam

inputkonventioner for gier - damspil.

<farve> ::= <sort>|<hvid>

spilleren taster hvilken farve han/hun ønsker at spille med.
hvid begynder altid.

<styrke> ::= <heltal>,

det tal der tastes, angiver det minimale niveau, som gier ser frem. et tal <1 vil blive opfattet som 1, og et tal >25 vil blive opfattet som 25. det gøres opmærksom på at programmets langsomhed stiger eksponentielt med styrkens størrelse. det tager ca 30 sek. at kigge 2 niveauer frem, og ca $3\frac{1}{2}$ min. at kigge 3 niveauer frem.

når udskriften din tur: kommer på skrivemaskinen forventes et træk af følgende struktur:

<træk> ::= <startpos> <mellempos> <slutpos>
<startpos> ::= <pos>,
<mellempos> ::= <empty> | <pos>, | <pos>, <mellempos>
<slutpos> ::= <pos>
<pos> ::= <bogstav> <højde>
<bogstav> ::= a | b | c | d | e | f | g | h
<højde> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8

notationen er sædvanlig skakbrætnotation. hvis man prøver at taste en position som ikke er korrekt, svares der med annul (i rødt), og hele trækket er annuleret, det samme sker hvis der tastes et å.

det kan anbefales, at man udstyrer sig med et dambræt, hvis man vil spille med gier, men har man ikke det vil kaon bevirke udskrift af et dambræt med de korrekte positioner. udskriften kommer hvergang gier har trukket. kbon bevirker en stackudskrift af programmets stack, samt en udskrift af de hidtil bedste træk.

```
   a,n<
   begin comment damspillende program;
 1    integer stacktop,ran,stp,oldtop,stinc,bufant,byte,maxniv,niveau,state,
 2    link,point,mi,t,i,ii,q,k,ant,alt,j,db,højde,bogst,goodnes,looki;
 3    integer array b[1:32,1:6];
 4    boolean newmove,quin,color,m,v,dm,di,slut,valg,look;
 5    boolean array buf,draw,a[0:10],stack[0:1000],hb[0:10,1:40];
 6
 7    procedure tostack;
 8    begin integeri;
 9       buf[bufant]:=newmove;
10       for i:=1 step 1 until bufant do
11       stack[stp+i-1]:=buf[i];
12       stack[stp+bufant]:=deadpos;
13       stack[stp+bufant+1]:=buf[0];
14       stack[stp+bufant+2]:=boolean(bufant+3);
15       stinc:=stinc+bufant+3;
16       stp:=stp+bufant+3;
17       alt:=alt+1
18    end tostack;
19
20    integer procedure look forward(color);
21    value color; boolean color;
22    begin integer i,ii,k,t;
23       boolean m,v;
24       bufant:=state:=1; stp:=stacktop+2;
25       link:=if color then 5 else 5;
26       alt:=stinc:=0;niveau:=niveau+1;
27
28       for i:=1 step 1 until 32 do
29       begin looki:=i;
30          if gier(look)=2 then
31          begin comment felt optaget af egen farve;
32             quin:=booleanb[i,1]shift2;
33             newmove:=buf[1]:=400v(booleani)v(quinshift8);
34             maxniv:=bufant:=1; buf[0]:=false; byte:=10;
35             move(i,false,-1,1);
36             b[i,1]:=integer(quinshift-2)
37          end gennemgang af trækmuligheder;
38       endfor;
39
40       if state=1 then
41       begin comment spillet tabt paa dette niveau;
42          newmove:=false;
43          buf[0]:=boolean(-100);
44          tostack
45       end;
46
47       stack[stacktop]:=booleanniveau;
48       stack[stacktop+1]:=booleanstinc;
49       oldtop:=stacktop;
50       stacktop:=stacktop+stinc+4;
51       stack[stacktop-2]:=false;
52       stack[stacktop-1]:=boolean(oldtop-1);
53       stack[oldtop-2]:=boolean(stacktop-2);
54       lookforward:=stacktop-2-(integerstack[stacktop-3])
55    end lookforward ;
56
```

```
 57     corecode look,looki,b,color;
 58     3,46
 59     2,44
 60     2,48
 61     2,46
 62     arna2
 63     hhreLZ
 64     xr,mlnre2
 65     xr,arc42
 66     ara3,i1
 67     pi(c17)
 68     hvnre1NOA
 69     arn5 12D
 70     ara4IOA
 71     hhnreLOC
 72     hhnreNOC
 73     e:arnc42,arc42
 74     e 1:arc42,hrs1
 75     e2:6
 76     e;
 77
 78
 79
 80     procedure  inmove(i,j);
 81     value i,j; integer i,j;
 82     comment brik flyttes fra felt nr.j til felt nr.i;
 83     begin  boolean bi1;
 84        bi1:=booleanb[i,1];
 85        if byte>30 then
 86        begin  byte:=0; buf[bufant]:=newmove;
 87           bufant:=bufant+1; newmove:=false
 88        end;
 89        newmove:=newmovev (( boolean i v( bi1 shift 10)) shift byte );
 90        byte:=byte+10;
 91        b[i,1]:=b[j,1]; b[j,1]:=0;
 92        if bi1 then
 93        point:=(if bi1 shift 2 then 10 else 4)  else point:=0;
 94        point:=point+ b[i,2] -  b[j,2]+(if(i>28vi<5)^(-,buf[1]shift-8) then 10
                                                  else0);
 95        buf[0]:= boolean ( integer buf[0]+point)
 96     end procedure inmove;
 97
 98     procedure reestab(i);
 99     value i;integer i;
100     comment brik flyttes i felter tilbage;
101     begin boolean c,old; integer i1,i2,j;
102        for j:=1 step 1 until i do
103        begin c:= (newmove shift (10-byte))^(30010m);
104           i1:= integer(c^3307m);
105           old:= (c^(3003m70)) shift -10;
106
107           b[i1,1]:= integer old;
108           newmove:=newmove ^(( 30m100) shift ( byte -  10));
109           byte :=byte - 10;
110           if byte= 0 then
111           begin byte := 40;
112              bufant := bufant - 1;
113              newmove := boolean buf[ bufant]
114           end;
```

```
115          i2 := integer(( newmove shift ( 10 - byte)) ^3307m);
116          if old then
117          point := ( if old shift 2 then 10 else 4) else point := 0;
118          point := point + b[ i1,2] - b[ i2,2]+(if(i1>28vi1<5)^(-,buf[1]shift
                                                    -8) then10else0);
119          buf[0] := boolean ( integer buf[0] - point )
120       end for loop
121    end procedure reestab;
122
123    procedure move(place,take,bad,niveau);
124    valueplace,take,bad,niveau;
125    booleantake;integerplace,bad,niveau;
126    begin integer j,sq,sqo;
127       if -,quin then
128       begin comment almindelig brik;
129          for j:=link,link+1 do
130          begin sq:=looki:=b[place, j];
131             case gier(look) of
132             begin
133
134                comment frit felt;
135                if -,take ^ state≤2 then
136                begin state:=2;
137                   inmove(sq,place); tostack;
138                   reestab(1)
139                end case1;
140
141                comment felt opt. af egen farve;;
142
143                comment felt opt. af modstander;
144                if state≤3 then
145                begin inmove(sq,place);
146                   sqo:=sq; sq:=looki:=b[sq, j];
147                   if gier(look)=1 then
148                   begin
149                      if state≤2 then
150                      begin stinc:=alt:=0; stp:=stacktop+2; state:=3end;
151                      inmove(sq,sqo);
152                      move(sq,true,bad,niveau+1);
153                      if maxniv<niveau then maxniv:=niveau;
154                      if maxniv=niveau then tostack else maxniv:=maxniv-1;
155                      reestab(2)
156                   end else reestab(1)
157                end case 3
158
159             end case
160          end
161       end almindelig brik else
162       begin comment dam;
163          for j:=3 step 1 until 6 do
164          if bad≠j then
165          begin sqo:=place; sq:=looki:=b[place,j];
166             try:
167             case gier(look) of
168             begin
169
170                begin comment frit felt;
171                   if -,take ^ state≤2 then
172                   begin state:=2;
173                      inmove(sq,sqo); tostack; reestab(1);
174                      sqo:=sq;
175                      for looki:=b[sq,j] while gier(look)=1 do
176                      begin sq:=looki; inmove(sq,sqo);tostack; reestab(1);
                                                    sqo:=sqend;
```

```
177                     sq:=looki
178                  end else
179                  begin sqo:=sq;
180                     for looki:=b[sq,j] while gier(look)=1 do sqo:=sq:=looki;
181                     sq:=looki
182                  end;
183                  goto try
184               end case1;
185
186               comment optaget af egen brik;;
187
188               begin comment optaget af modstander;
189                  inmove(sq,sqo);
190                  sqo:=sq; sq:=looki:=b[sq,j];
191                  if gier(look)=1 then
192                  begin
193                     if state≤3 then
194                     begin stinc:=alt:=0;stp:=stacktop+2;state:=4end;
195                     inmove(sq,sqo);
196                     move(sq,true,(j-1)mod4+3,niveau+1);
197                     if maxniv<niveau then maxniv:=niveau;
198                     if maxniv=niveau then tostack else maxniv:=maxniv-1;
199                     reestab(2)
200                  end elsereestab(1)
201               end case 3
202
203            end case
204         end
205      end dam
206   end procedure move;
207
208   boolean procedure comp;
209   comment comp sammenligner et træk i stakken med hidtil bedste (hb)
210   og har værdien true, hvis der er flere træk tilbage og false ellers.
211   desuden reetablerer comp dambrættet;
212
213   begin integer i,ii,j,k,c,niveau,p1,p2,ant;
214      boolean boo,boo1;
215      boo1:=true;
216      et mere:
217      i:=integerstack[1];
218      ant:=integerstack[i-1]-1;
219      k:=i-ant-4;
220      for j:=3 step 1 until ant do
221      a[j]:=stack[k+j];
222      niveau:=p1:=0; boo:=true;
223      for i:=i while i≠0 do
224      begin comment pointsammentælling ved linkning gennem stakken;
225         p1:=p1+(if boo then integer stack[i-2] else -integerstack[i-2]);
226         ii:=i; i:=integerstack[i];
227         boo:=-,boo;
228         niveau:=niveau+1
229      end pointsammentællingfor;
230      a[1]:=booleanniveau;
231      a[2]:=booleanp1;
232      if integerhb[1,niveau]=0 then goto gem else
233      begin p2:=integerhb[2,niveau];
234         if(boo^p1<p2)v(-,boo^p2<p1)v(p1=p2^random)then
235         gem:
236         begin
237            for j:=1 step 1 until ant do
238            hb[j,niveau]:=a[j];hb[j,niveau]:=false
239         end
240      end;
241
```

```
242         vrkd:
243         ant:=integerstack[ii-1];
244         j:=ii-ant;
245         k:=ii-4;
246         if bool then move back(stack,j,k);
247         c:=oldtop-1;
248         bool:=c+3=j;
249         if bool^oldtop=3 then
250         begin comp:=true; goto exit end;
251         if ¬ bool then
252         begin stack[c-1]:=booleanj;
253            stack[j]:=false;
254            stack[j+1]:=booleanc;
255            stack[c+2]:=boolean(integer(stack[c+2])-ant);
256            stacktop:=stacktop-ant;
257            if stack[stacktop-5] then goto et mere
258         end flere træk tilbage paa dette niveau else
259         begin stack[c-1]:=false;
260            stacktop:=oldtop;
261            oldtop:=integerstack[c]+1;
262            c:=integerstack[oldtop];
263            niveau:=c+1;
264            ifintegerhb[1,c]≠c then goto gem1 else
265            begin boo:=(c mod 2)=1;
266               p1:=integerhb[2,c]; p2:=integerhb[2,niveau];
267               if(boo^p1<p2)∨(¬boo^p2<p1)∨(p1=p2^random)then
268               gem1:
269               begin i:=1;
270                  hb[1,c]:=booleanc;
271                  for i:=i+1 while i<3 ∨integerhb[i,niveau]≠0 do
272                  hb[i,c]:=hb[i,niveau];hb[i,c]:=false
273               end;hb[1,niveau]:=false
274            end;
275            ii:=stacktop-2;
276            goto vrkd
277         end ikke flere træk paa dette niveau;
278         comp:=false;
279         ant:=integerstack[stacktop-3]-3;
280         ii:= stacktop-5-ant;
281         k:=ii+ant-1;
282         move ahead(stack,ii,k);
283         exit:
284      end comp;
285
286      booleanprocedure random;
287      code ran;
288      2,44
289      pma1,mlnre
290      gma1,sca1
291      arna1,ck1
292      grp-1,hvr2
293      e:990 1
294      e;
295
```

```
296     integer procedure readkoord;
297     codei;
298     2,44
299     panre1t58ITA
300     sy58,
301     hsre
302     hvre7LTA
303     is-57
304     bss,hvre3
305     sr48D
306     ck10,grre4
307     ck-11OA
308     ck1,src42
309     hvre3LT
310     hsre
311     hvre7LTA
312     is-10
313     bss,hvre3
314     ck9IOB
315     ck1,pmre4
316     grre4,arnre5
317     srre4X
318     mlre5X
319     arc42NOB
320     ck-1,grp-1
321     hvre3LOC
322     hvre3NOC
323     arnre1,ca60
324     e3:srnc42,grp-1
325     hsre
326     e7:ck10,grc54
327     hvre8
328     e1:qq
329     e5:8
330     e4:qq
331     e:lyns3,ca13
332     srnc42ITA
333     nc60,ca58
334     grre1,hvre
335     hvreLZ
336     hrs1
337     e8:e;
338
339     procedure printboard;
340     begin inteher bi,i,j,sel;
341        boolean s1,s2,s3,boo;
342        sel:=select(8);
343        boo:=false; s3:=4106103 00;
344        s1:=4106106446446446446 44;
345        s2:=4106106 106106106106 10;
346        writecr;writecr;writechar(30);
347        writechar(9);writetext(*<HVID*);
348        writecr;writecr;
349        for i:=32 step -4 until 2 do
350        begin writetext(ifboothenstrings1elsestrings2);
351           for j:=0 step 1 until 3 do
352           begin bi:=integer(booleanb[i-j,1]shift3)+1;
353              writetext(casebiof(strings3,*< gal *,*< gal *,*< gal *,
354              *<₁₁₁₁₁*,*<₁₁s₁₁*,*<₁₁₁₁₁*,*<₁₁h₁₁*));
355              ifj<3thenwritetext(strings1)
356           end;
357           ifboothenwritecr;
358           writetext(strings1);if-,boothenwritecr;
```

```
359          for j:=0 step 1 until 3 do
360          begin bi:=integer(booleanb[i-j,1]shift3)+1;
361             writetext(casebiof(strings3,*< gal >*,*< gal >*,*< gal >*,
362             *<11s11>*,*<1sss1>*,*<11h11>*,*<1hhh1>*));
363             if j<3thenwritetext(strings1);
364          end;if-,boothenwritetext(strings1);
365          writeinteger(*<1dd>*,9-(i:4));writecr;
366          ifboothenwritetext(strings1);
367          for j:=0 step 1 until 3 do
368          begin bi:=integer(booleanb[i-j,1]shift3)+1;
369             writetext(casebiof(strings3,*< gal >*,*< gal>*,*< gal >*,
370             *<111111>*,*<11s11>*,*<111111>*,*<11h11>*));
371             ifj<3thenwritetext(strings1)
372          end;if-,boothenwritetext(strings1);
373          writecr;boo:=-,boo
374       end;writecr;
375       writetext(*<11>*);
376       forj:=8step-1until 1do
377       beginwritechar(48+j);writetext(*<1111>*)end;
378       writecr;writecr;writechar(30);
379       writechar(9);writetext(*<sort>*);writechar(66);
380       select(sel)
381    end printboard;
382
383    procedure move back(a,i1,i2);
384    valuei1,i2;integeri1,i2;
385    boolean array a;
386    begin boolean move;integeri,v,t;
387       for i:=i1 step 1 until i2 do
388       begin move:=a[i];
389          for i:=i while integermove≠0 do
390          begin t:=integer(move^3307m);
391             v:=integer((moveshift-10)^3m370);
392             move:=(moveshift-10)^10030m;b[t,1]:=v
393          end
394       endfor
395    end move back;
396
397    procedure move ahead(a,i1,i2);
398    value i1,i2; integer i1,i2;
399    boolean array a;
400    begin boolean move;integeri,v,t;
401       v:=integer((a[i1]^3003m70)shift-10);
402       for i:=i1 step 1 until i2 do
403       begin
404          move:=a[i];
405          t:=integer(move^3307m);
406          move:=move^30m100shift-10;
407          for i:=i while integermove≠0 do
408          begin b[t,1]:=0;
409             t:=integer(move^3307m);
410             move:=move^30m100shift-10
411          endfor;b[t,1]:=0
412       end;
413       if t>28vt<5 then v:=integer(booleanvv201 1330);
414       if t≤0vt>32then begin printstack;goto exitend;
415       b[t,1]:=v
416    end move ahead;
417
```

```
418     boolean procedure deadpos;
419     deadpos:=((state=2^niveau≥goodnes)vstate=1)vniveau>25;
420
421
422     procedure printstack;
423     begininteger i,ii,iii,l,tabul,sel,k,j,t,højde,bogst;
424        boolean move;
425        sel:=select(8); i:=ii:=1;writetext(¥<stackudskrift:¥);
426        writecr;writecr;
427        for i:=i  while i≠0 do
428        begin tabul:=2;
429           k:=ifintegerstack[i]=0then1else3;
430           for j:=0step1until1kdo
431           beginwriteinteger(¥  -ddd¥,integerstack[i+j]);writecrend;
432           i:=integerstack[i]; iii:=i;
433           if i=0 then goto exit;
434           for i:=i while i≠ii+4 do
435           begin k:=i-4;tabul:=tabul+3;
436              writechar(30);writechar(tabul);
437              i:=i-integerstack[i-1];
438              for j:=i step 1 until k do
439              begin move:=stack[j];
440                 fori:=i while integer move≠0 do
441                 begin t:=integer(move^330Zm);
442                    højde:=(t-1):4+1;
443                    bogst:=48+(2×t-1)mod8 +(højde+1)mod2;
444                    writechar(bogst);writechar(9-højde);writechar(27);
445                    move:=move^30m100shift-10
446                 end
447              end;
448              for l:=1 step 1 until 3 do
449              begin
450                 writeinteger(¥  -ddd¥,if1=1then(ifstack[k+1]then1else0)elseint
451              end;writecr
452           end;writecr;writecr;writecr;i:=ii:=iii
453        end;
454        exit:
455        writecr;writecr;writecr;
456        writetext(¥<udskrift af sammenligningsstack:¥);writecr;
457        writecr;writetext(¥<  niveau   point    træk¥);writecr;
458        forj:=1step1until39do
459        if integerhb[1,j]=jthen
460        begin write(¥    -ddd¥,integerhb[1,j],integerhb[2,j]);
461           writetext(¥<₁₁₁₁¥);
462           move:=hb[3,j];
463           for i:=4,i+1while integer move≠0 do
464           beginfor i:=i while integer move≠0 do
465              begin  t:=integer(move^330Zm);
466                 højde:=(t-1):4+1;
467                 bogst:=48+(2×t-1)mod8+(højde+1)mod2;
468                 writechar(bogst);writechar(9-højde);writechar(27);
469                 move:=move^30m100shift-10
470              end;move:=hb[i,j]
471           end;writecr
472        end;writechar(66);
473        select(sel)
474     end printstack;
475
476
```

```
11:if where(¢<dambræt¢,db)¢0 then goto 11;
15:ifget(hb,db,1)<0thengoto15;
hb[10,40]:=boolean(integerhb[10,40]-1);
ran:=integerhb[10,40];
16:ifput(hb,db,1)<0thengoto16;
12:if get(b,db,1)< 0 then goto 12;
select(8);writechar(66);
writetext(¢<   inputkonventioner for gier - damspil:

<farve>         ::=<sort>|<hvid>

spilleren taster hvilken farve brikker han
ønsker at spille med. hvid begynder altid.

<styrke>        ::= <heltal>,

det tal der tastes angiver det minimale an-
tal træk, som gier ser frem. tal ≤1 opfattes
som 1, og tal ≥25 opfattes som 25. med styrke
lig 2 tager det ca. 15 sek. for gier at trække,
og med styrke lig 3 tager det ca. 2 min.  nor-
malt vil styrke 3 give passende modstand.

<din tur:>      ::= <startpos> <mellempos> <slutpos><
<startpos>      ::= <pos>,
<mellempos>     ::=  |<pos>|<pos>,<mellempos>
<slutpos>       ::= <pos>
<pos>           ::= <bogstav><højde>
<bogstav>       ::= a|b|c|d|e|f|g|h
<højde>         ::= 1|2|3|4|5|6|7|8

notationen er sædvanlig skakbrætnotation. hvis
man prøver at taste en position ind som ikke er
korrekt, svares der med annul, og hele trækket
er annuleret. desuden kan annulering ske efter
sædvanlig help 3 konvention.

det kan anbefales at udstyre sig med et dambræt naar man vil
spille med gier, men har man ikke det, vil kaon bevirke udskrift
af et dambræt paa linieskriveren, hver gang gier har trukket.¢);
writechar(66);printboard;
select(17);writecr;fori:=1stepluntil40dohb[1,i]:=false;
writetext(¢<GIER - DAMSPIL¢);writecr;

14:
writecr;
writetext(¢<farvevalg: ¢);j:=lyn;
if j=18 then
begin writetext(¢<ort¢); valg:=false end else if j=56 then
begin writetext(¢<vid¢); valg:=true end else goto 14;
writecr;writetext(¢<styrke: ¢);goodnes:=readinteger;
if-, valg then goto os;
start:
color:=valg;
stacktop:=3;
stack[1] := 39 0 11;
oldtop:= stacktop;
stack[2] := false;
niveau := 0;
lookforward(color);
if state=1 then
begin writetext(¢<jeg vinder dette spil¢); goto exitend;
```

```
539     om igen:
540     i:=1; dm:=false; q:=0;
541     writecr; writetext(‡<din tur  ‡);
542     comment indlæsning af modstanders træk;
543     char:=0;
544     for k:=k while char ‡17 do
545     begin k:=readkoord;
546        if char<0∨k>32∨k≤0 then
547        begin writecr;
548           for i:= 29,49,37,37,20,35,62 do
549           writechar(i); goto om igen
550        end;
551        if q=40 then
552        begin q:=0;draw[i]:=dm;
553           dm:= false; i:=i+1
554        end;
555        dm:=dm∨(((booleank)∨(booleanb[k,1]shift10))shiftq);
556        q:=q+10;
557     end;
558     draw[i]:=dm;
559     draw[i+1]:= false; comment  slut indlæsning af træk;
560     comment kontrol af modstanders træk;
561     mi:=i; ii:=stacktop-3;
562     nyttræk:
563     ant:=integerstack[ii]-3;
564     ii:=ii-ant-3;
565     if mi‡ant then
566     next:
567     begin if ii=4 then
568        begin writecr;
569           writetext(‡<e j lovligt træk - prøv igen‡);
570           goto omigen
571        end;
572        goto nyt træk
573     end;
574     di:=draw[1];
575     for i:=1 step 1 until ant do
576     if integer di‡integerstack[i+ii] then gotonext
577     else di:=draw[i+1];
578
579     comment modstanderens træk trækkes;
580     move ahead(draw,1,mi);
581
582     comment vi begynder at se os om efter et modtræk;
583     os:stacktop:=oldtop:=3;
584     stack[1]:=3901 1;stack[2]:=false;niveau:=0;
585     color:=-,valg;
586     ii:=look forward(color);
587     if state=1 then
588     begin writetext(‡<du vinder dette spil‡); goto exit end;
589     if alt=1 then
590     begin for i:=ii step 1 until stacktop-6 do
591        hb[i-ii+3,1]:=stack[i];
592        hb[i-ii+3,1]:=false
593     end else
594     begin
595        move ahead(stack,ii,stacktop-6);
596        slut:=false;
597        for i:=i while -,slut do
598        begin for i:=i while -,dead pos do
599           begin color:=(stack[oldtop]shift-1)=valg;
600              ii:=look forward(color);
601              if state>1 then move ahead(stack,ii,stacktop-6);
602           endfor;ifkbonthenprintstack;
603           slut:=comp; niveau:=integerstack[oldtop];
604
```

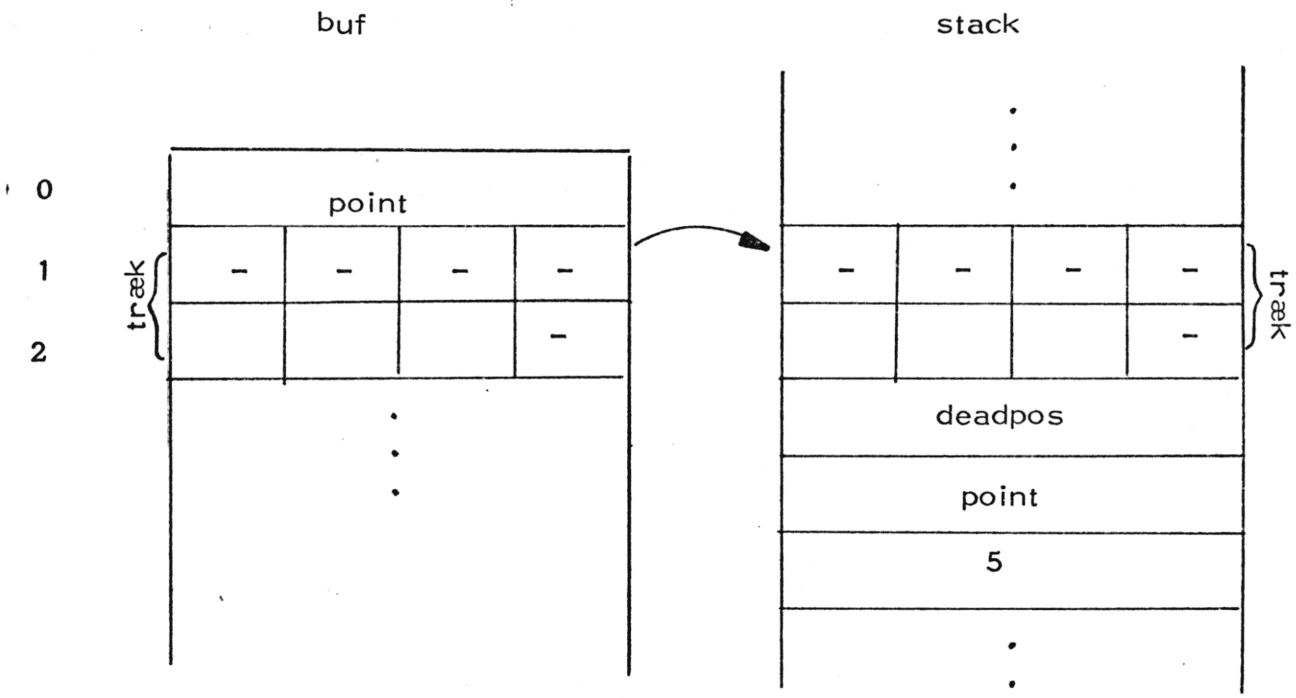
```
604             state:=ifstack[stacktop-5]then1else3
605          end
606       end;
607       comment paa dette sted i programmet skulle vi
608       have fundet det bedste træk,som derfor lægges ind nu;
609       writecr;writetext(*<jeg trækker *);
610       hb[1,1]:=false;
611       i:=2; m:=hb[3,1];
612       v:=(mshift-10)^3m370;
613       t:=integer(m^3307m);
614       højde:=((integert)-1):4+1;
615       bogst:=48+((2×(integert))-1)mod8+(højde+1)mod2;
616       writechar(bogst);writechar(9-højde);
617       m:=(mshift-10)^10030m;
618       fori:=i+1 while integerm‡0 do
619       begin for i:=i while integerm‡0 do
620          beginb[t,1]:=0;
621             t:=integer(m^3307m);
622             højde:=((integert)-1):4+1;
623             bogst:=48+((2×(integert))-1)mod8+(højde+1)mod2;
624             writechar(27);writechar(bogst);
625             writechar(9-højde);
626             m:=(mshift-10)^10030m;
627          end;
628          m:=hb[i+1,1]
629       end;writechar(17);
630       if t<5vt>28 then v:=vv2011330;
631       if t‚<1vt>32 then begin printstack; goto exit end;
632       b[t,1]:=integerv;
633       ifkaonthenprintboard;
634       goto start;
635
636       exit:
637       writecr; writetext(*<skal vi prøve igen*);j:=lyn;
638       ifj=33thenbeginwritechar(49);writecr;goto12endelseifj=37then
639       beginwritechar(53);writechar(33);writecrendelse gotoexit;
640
641       writetext(*<tak for spillet*);writecr
642    endt<
```

GENNEMGANG AF PROGRAMMET.

linie 7-18.

tostack er en procedure, som lægger indholdet af et array "buf" over i sammenligningsstakken "stack" således:
først overføres selve trækket til stakken, dernæst kommer en celle, der indeholder en boolsk værdi "deadpos", som er knyttet til dette træk, og som angiver hvorvidt det er interessant at udvikle dette træk yderligere. Dernæst en celle indeholdende trækkets pointtal efterfulgt af en celle, der angiver, hvor mange stackelementer, der ialt er brugt i sammenligningsstakken på dette niveau.

EKSEMPEL:



linie 20-55

proceduren look forward gennemgår alle felterne på brættet, og opdaterer tilsidst pointere i stakken.

linie 57-76

core code look

ender ud med en værdi i r-registret, som fortæller indholdet af felt nr. looki på brættet. indholdet i r-registret er:

1 hvis feltet er frit

2 hvis feltet er optaget af egen brik, eller der ikke er noget felt.

3 hvis feltet er optaget af modstander.

proceduren er kodet i slip fordi den bruges meget tit, og vi ville derfor gerne have den "oppe" hele tiden.

Linie 80-96.

Inmove(i, j) er en procedure som
i) flytter en brik fra felt nr j til felt nr i, og som
ii) foretager pointsammentælling.

Et felt er repræsenteret ved en 10-bit gruppe, hvoraf de tre første bit indeholder b[i, 1] og de resterende syv bit heltallet "i".

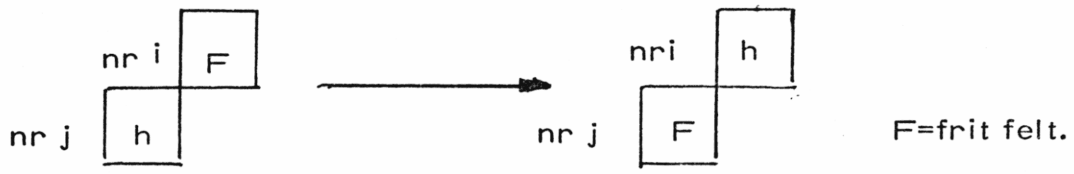
Internt benytter vi os af en hjælpecelle "newmove", hvortil der er knyttet en pointer "byte", som holder øje med hvilken 10-bit gruppe, vi stårved, samt et array "buf" med "bufant"som pointer.

Ethvert felt placeres først i newmove. Når denne så indeholder fire felter dvs byte = 40, lægges newmove over i arrayelementet buf[bufant].

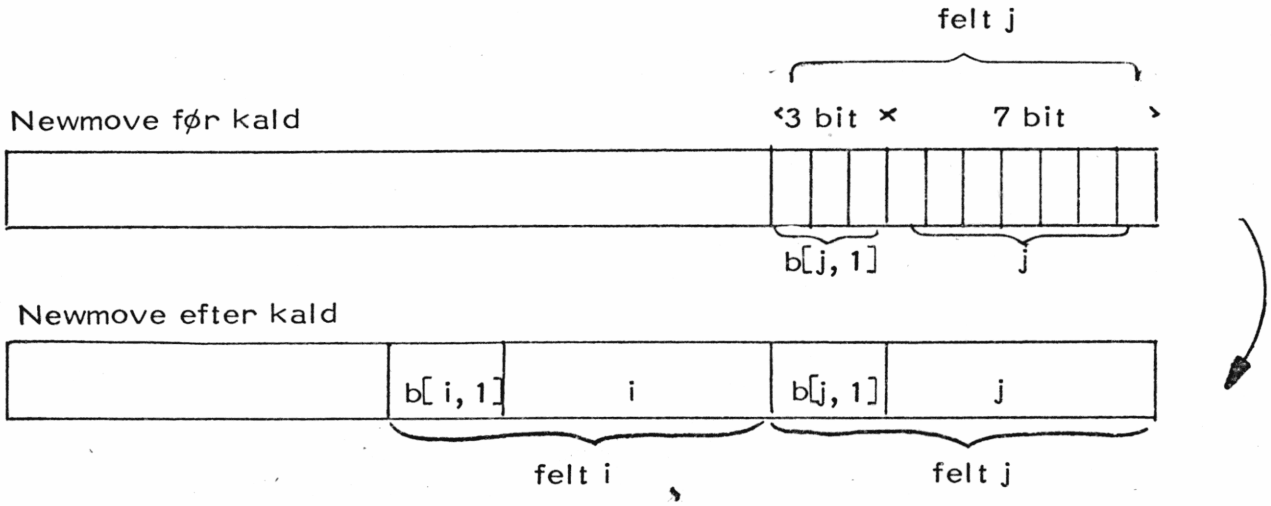
Med andre ord "buf" opbevarer ét lovligt træk, før det overflyttes til stakken "stack" ved proceduren "tostack".

EKSEMPEL:

EKSTERNT:



INTERNT:



Pointsammentælling for et træk:

a) Man får 10 point for at slå eller opnå en dam.
b) Man får 4 point for at slå en alm. brik.
c) Man får point for blot at flytte fra felt nr j til felt nr i, nemlig værdien af felt nr i minus værdien af felt nr j.
Internt er det udtrykt ved b[i, 2] - b[j, 2].

Det samlede pointtal for et træk placeres i buf[0].

Linie 98-121.

Reestab(i) er en procedure, der, som navnet angiver, reetablerer en tidligere situation. Heltallet "i" angiver, at "i" felter reetableres på brættet.

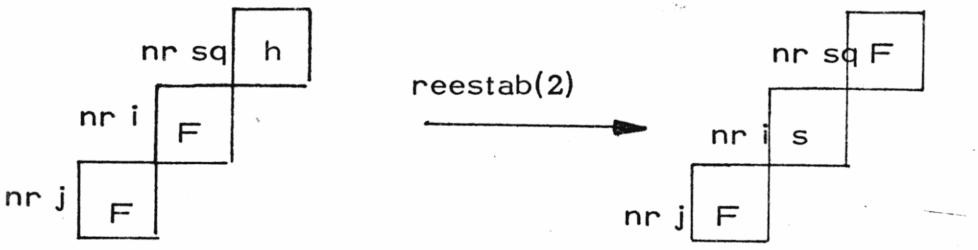
Desuden sker der en nedtælling i pointtallet.

Eftersom inmove og reestab er hinandens inverse, er der ingen grund til at gå i detaljer vedrørende reestab.

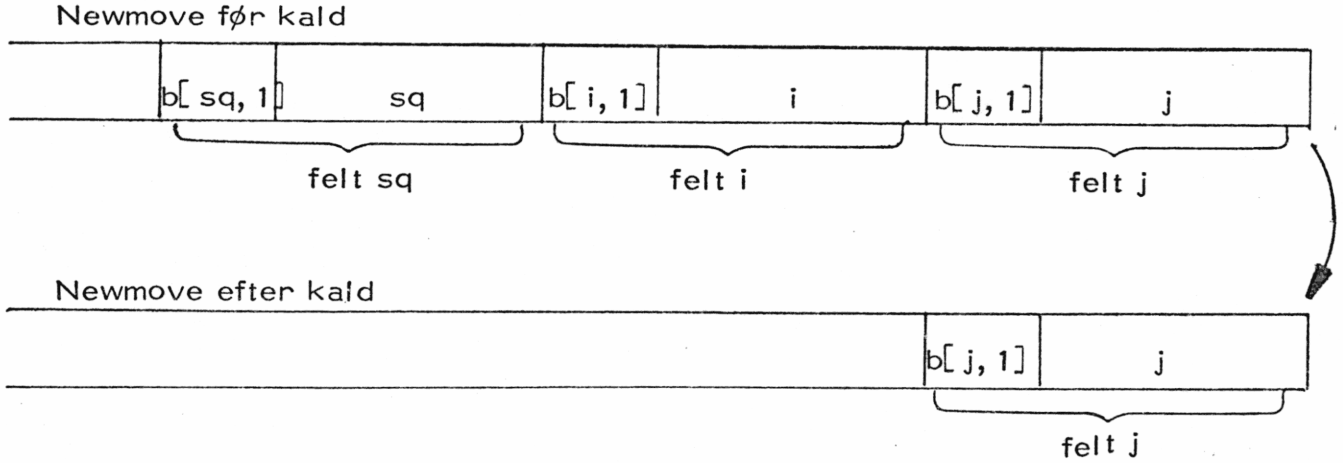
Inmove opbygger, reestab river ned.

EKSEMPEL:

EKSTERNT:



INTERNT:



Linie 123-206.

Move(place, take, bad, niveau) er en recursiv procedure, som i en stak opbygger alle de slag-eller trækmuligheder en brik, der befinder sig på feltet med nr "place" , har. Move har indbygget en tilstandstabel. Denne er udtrykt ved variablen "state". Betydningen af state er som følger:

state = 1 ⇄ ingen brik kan hverken flytte eller slå.
state = 2 ⇄ en brik kan flytte men ikke slå.
state = 3 ⇄ en alm. brik kan slå.
state = 4 ⇄ en dam kan slå.

Hvis state noget øjeblik antager værdien q (q=3 ∨ q=4), vil alle slag eller træk svarende til state < q blive afstakket ligesom intet træk med state < q vil blive stakket, svarende til en gevinst i søgetid og plads.

"take" er en boolsk variabel, der holder øje med om den pågældende brik har slået eller ej.

"niveau" er en integer variabel, der fortæller os på hvilket niveau den recursive procedure er.

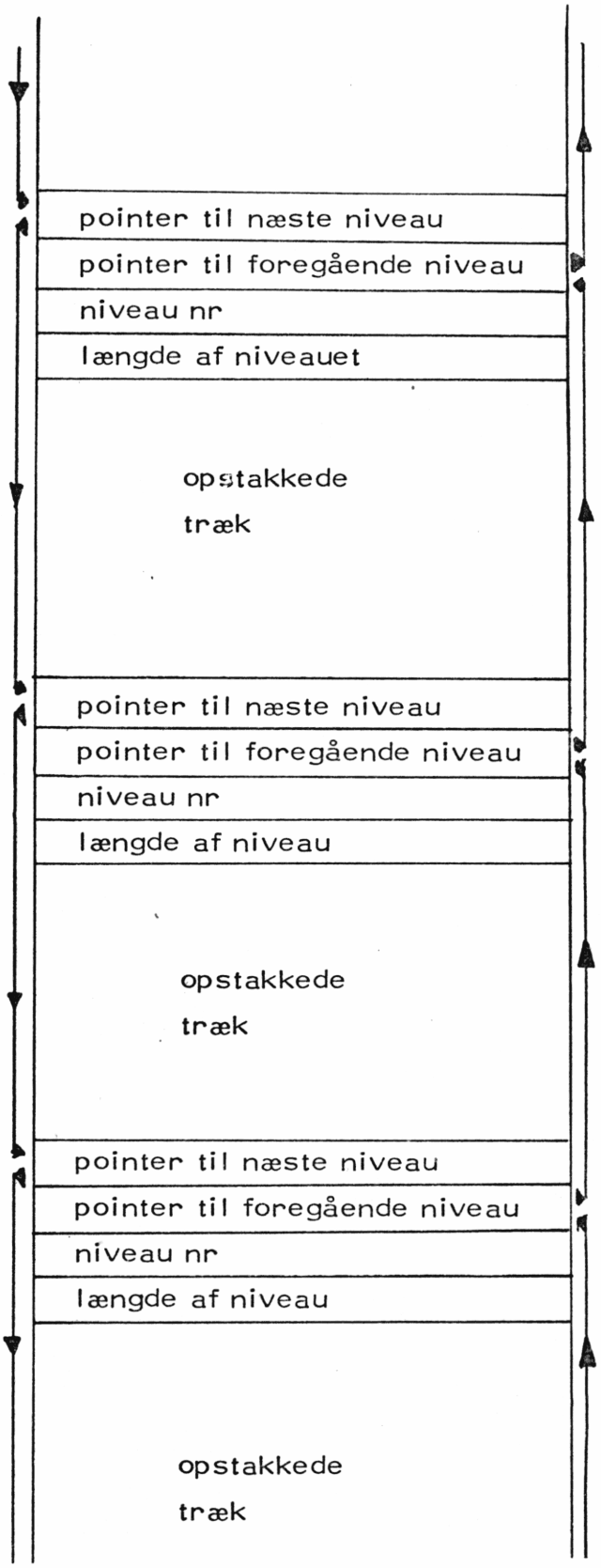
"bad" er en integer variabel, der sørger for at en dam ikke foretager en ulovlig slagserie. Har den f. eks. slå - et en brik i en bestemt retning, må den ikke umiddelbart derefter løbe tilbage i samme retning og der slå en anden brik.

prioriteringen af trækkene i dam er implimenteret i nedenstående tilstandstabel. tabellen benyttes i look forward og move.

	kan intet	kan flytte	kan slå med alm.	kan slå med dam
state 1:	goto 1	alt:=1 tostack goto 2	alt:=1 tostack goto 3	alt:=1 tostack goto 4
state 2:	goto 2	alt:=alt+1 tostack goto 2	alt:=1 tostack goto 3	alt:=1 tostack goto 4
state 3:	goto 3	goto 3	alt:=alt+1 tostack goto 3	alt:=1 tostack goto 4
state 4:	goto 4	goto 4	goto 4	alt:=alt+1 tostack goto 4

linie 208-284. proceduren comp.

for at finde det bedste træk benyttes en metode som kaldes min-max metoden. denne metode vil blive omtalt senere.
de dele af spilletræet som vi opbygger, opbevares i en stak eller rettere et system af stakke. et niveau i stakken består således af alle de træk en spiller kan gøre når det er hans tur til at trække, en trækserie består så af sidste træk på alle niveau'er. stakken er opbygget som vist herunder.



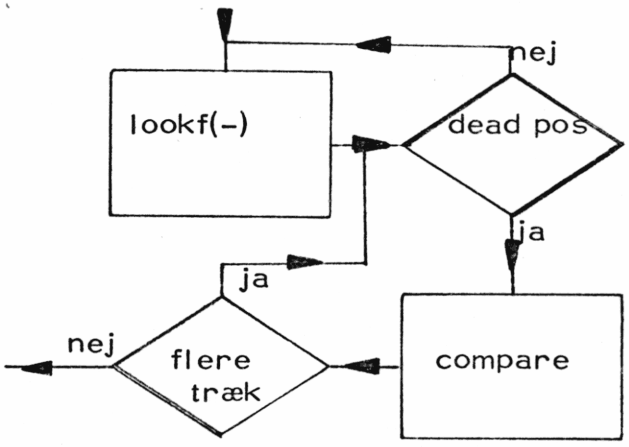
de opstakkede træk er af følgende udseende:

positioner i 10-bit grupper
deadposition
point for trækket
antal celler

ved at benytte en stak, istedet for at bygge hele træet op, begrænser vi lagerkravet ganske betydeligt, med et lagerkrav på kun 1000 celler kan vi således se ca. 15 træk frem for hver spiller.

MIN -MAX metoden.
hvert træk en spiller kan gøre får et pointtal. Da det altid vil være giers træk der står først i stakken, har vi valgt at vurdere en trækserie i forhold til gier (comp linie 223 - 229). da det også er således at den spiller der har initiativet, må forventes at gøre det beste træk, han/hun kan, må vi, når vi befinder os på et ulige niveau vælge det træk, som giver det højeste pointtal, og når vi befinder os på et lige niveau, vælge det træk, som giver det laveste pointtal. det beste træk gemmes. i uheldigste tilfælde kan vi blive nødt til at gemme lige så mange træk, som vi har niveauer. når alle træk på et niveau er vurderet, og det beste træk fundet, skal dette træk regnes for at tilhøre et niveau lavere, og sammenligning med et eventuelt tidligere gemt træk på dette niveau kan finde sted (comp linie 259 - 277).

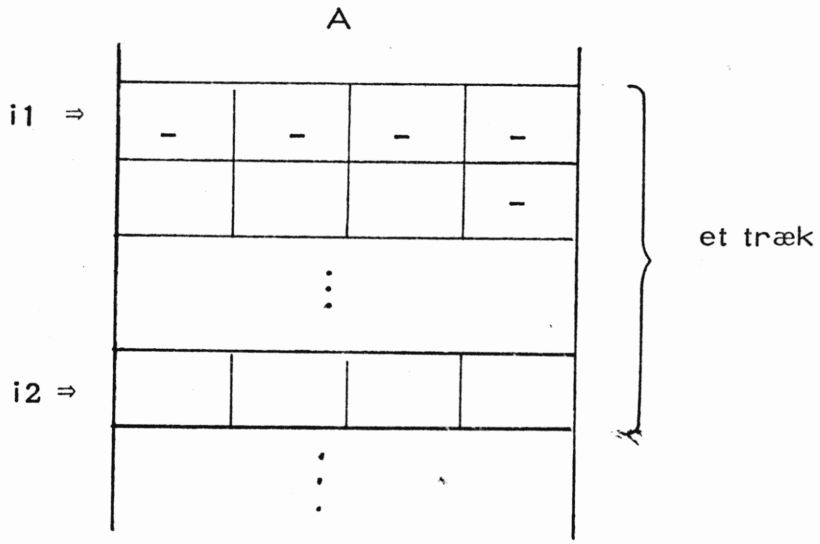
foruden at være den procedure der tager sig af sammenligning, er comp også den procedure der tager sig af nedbringningen af stakken, mens look forward tilføjer nye niveauer, når alle trækmuligheder er vurderet får comp værdien true, og gier gør sit træk.



linie 286-294 random benyttes når to træk er lige gode.

linie 296-337
readkoord transfornerer fra koordinater i skakbrætformat til intern repræsentation. kodning i slip er gjort for at kunne følge med skrivemaskinen.

Linie 383-395.



Moveback(A, i1, i2) flytter det træk, der står i cellerne A[i 1], . . . , A[1 2], tilbage på brættet.

Linie 397-416.

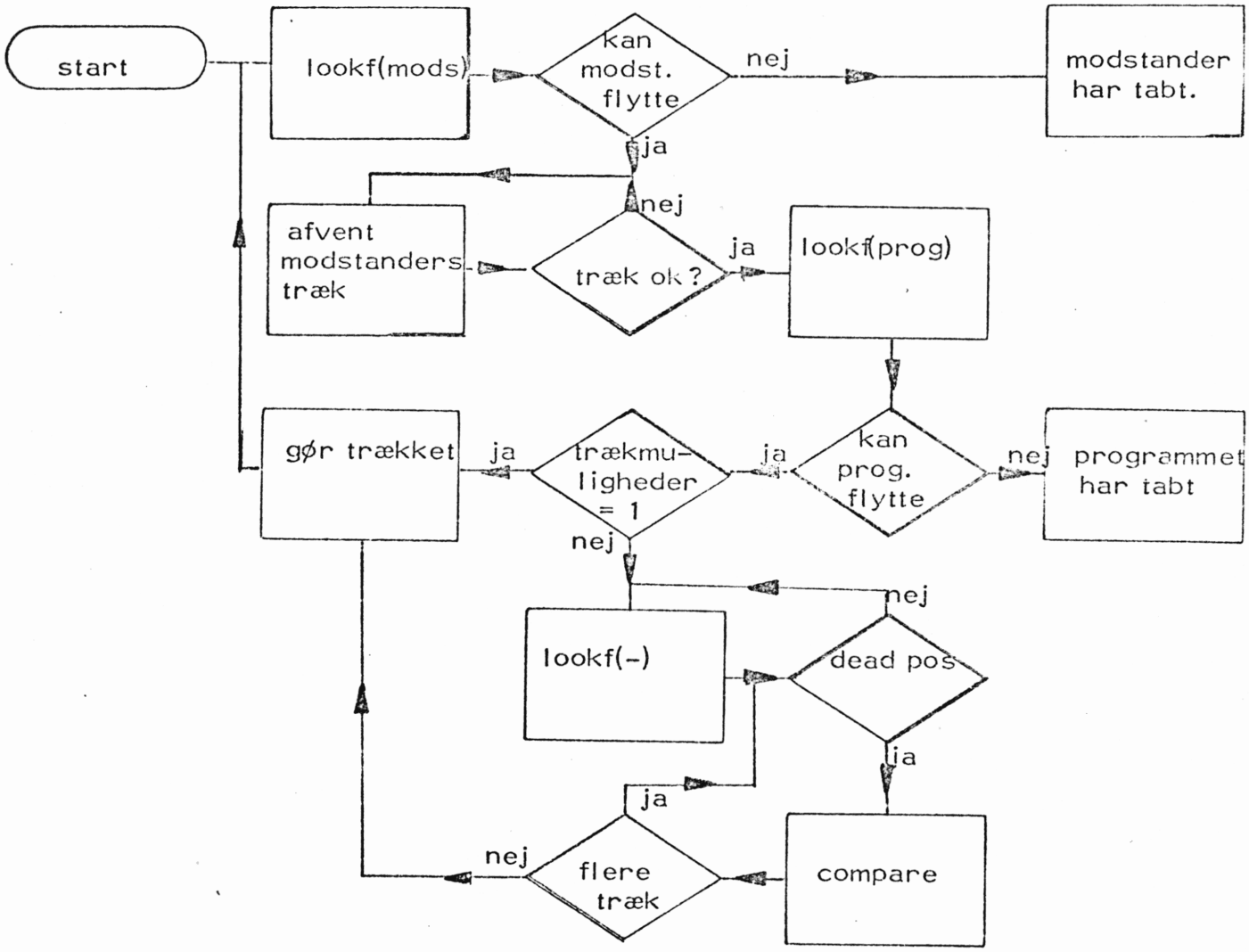
Se ovenstående fig.

Moveahead(A, i1, i2) foretager det træk, der står i cellerne A[i1], . . . A[i2], på brættet og sørger specielt for etableringen af en dam (se linie 413).

Moveahead og moveback er hinandens inverse.

linie 422-474
printstack giver et øjebliksbillede af "stack" og "hb" . udskriften fås ved at tænde kb. proceduren blev hovedsagelig lavet for at lette indkøringen af programmet.

Linie 477-642. flowchart.



litteraturliste:

E. A. Feigenbaum	Computers and thought.
A. L. Samuel	Some studies in machine learning using the game of checkers. II-recent
A. Elithorn	
A. Telford	Game and problem structure in relation to the study of human and artificial intelligence.
D. Michie	Game-playing and game-learning automata.
B. A. Trakhtenbrot	Algorithm and automatic computing machines.