IT3105 - 2048

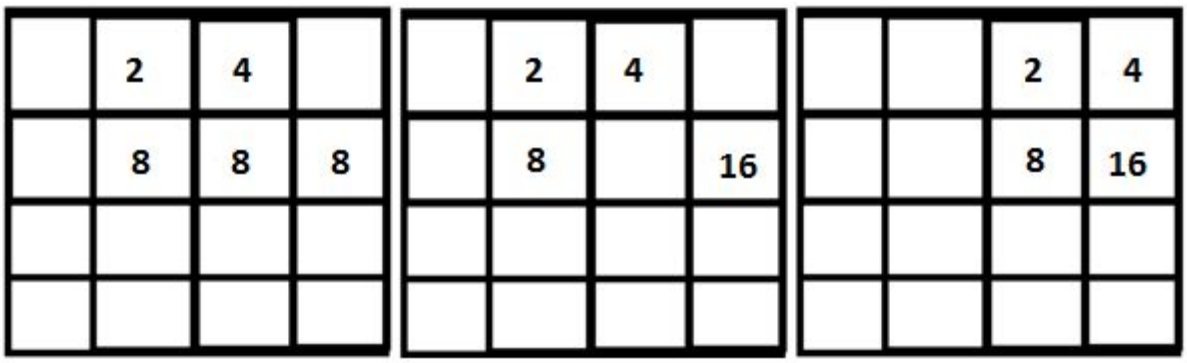
Bjørnar Remmen og Joakim Andal

Generell struktur:

Vi har valgt å bruke en todimensjonal tabell for å representere brettet gjennom spillet. Vi valgte også å bruke Java for at det skulle være noe raskere enn ved bruk av for eksempel Python.

1) **Describes the overall structure and key elements of your Expectimax or Minimax-with-alpha-betapruning system.**

For å kunne bevege brikkene rett har vi valgt å gjøre dette ved å først lage en metode som kombinerer de rette rutene og flytter deretter brikkene. Den beste retningen blir så oppdatert på det grafiske brukergrensesnittet.
Her vist med en bevegelse i høyre retning.



I vårt prosjekt har vi vært gjennom mange ulike løsninger før vi kom til den endelige løsningen som nå er expectimax. Vi har implementerst minimax, med og uten alpha-beta pruning, og expectimax. Grunnen til at expectimax ble implementert var at vi først ikke fikk minimax til å fungere grunnet et stor feil i spillet. Denne feilen ble først rettet opp etter at vi hadde implementert alle de tre forskjellige løsningene. Vi prøvde deretter minimax med alpha-beta pruning og expectimax, og fant ut at expectimax hadde det beste resulatet med heuristikkene vi prøvde ut.

I vår expectimax har vi representert hver state som en SearchNode, som har tilstanden til brettet og kan regne ut heuristikken fra dette objektet. Vår expectimax er delt inn i to deliterasjoner, der den ene iterasjonen er når “spillleren”, eller “AI-en”, velger en bevegelse som er enten opp, ned, høyre eller venstre. Den andre deliterasjonen er for det tilfeldige elementet i spillet, alstå en ny rute med enten en toer eller firer kommer til. Begge deliterasjonene er rekursive og kaller hverandre annen hver gang. Den første deliterasjonen returnerer den beste heuristikken funnet, og den andre returnerer gjennomsnittsverdien til bordet av alle de mulige kombinasjonene av de tilfeldige tallene.

Hver deliterasjon returnerer et objekt som har både SearchNode og en verdi som er heuristikken eller gjennomsnittsverdien av heuristikken til bordet. Hovedgrunnen til at expectimax også returnerer SearchNode er at man lett kan hente ut hvilke bevegelser som er brukt, og da også hente ut den bevegelsen som hadde høyest heuristikkpoeng.

2) **Clearly documents (in full detail) your heuristic function. This must NOT be presented as code, but rather as a mathematical expression where every symbol is clearly defined.**

Vår heuristikk bruker mye forskjellig. En av de tingene vi bruker er en vektet matrise(venstre) multiplisert med den aktuelle tilstanden representert som 2d matrise. Matrisen blir flippet rundt 90, 180 og 270 grader slik at vi har fire matriser. Disse matrisene blir også speilet slik at vi sitter igjen med åtte forskjellige matriser. Dette gjør at man altså ikke må spille i mot et spesielt hjørne, men at man har frihet til å velge hvilket som er best. Vi tar da å ganger slik som under og bruker den matrisen som ga høyest verdi. Dette skal guide litt hvordan de ulike brikkene skal plassere seg.

65536	32768	8192	16384
512	1024	2048	4096
32	64	128	256
2	4	8	16

*

32	256	124	0
512	64	0	2
0	0	0	0
0	0	0	0

=

(65536*32)/ 10 000	(32768*256 /10 000	(8192* 124)/ 10 000	0
(512*512 /10 000	(1024 *64) / 10 000	0	(4096*2) / 10 000
0	0	0	0
0	0	0	0

10 000

=

209.71	838.86	101.58	0
26.21	6.55	0	0.81
0	0	0	0
0	0	0	0

Dette blir $\sum$(array) = 1183.72

Vi ganger så summen med 2.2 Etter dette sjekker vi om vi har de største verdiene i i hjørnene, og om det er tilfellet ganger vi summen med 1.7

Etter dette har vi en måte for å sjekke om man har like verdier nært hverandre (clustering score/ spredt sum). Den måler om ting er spredt eller ikke. Like verdier som naboer er lett å kombinere og får da lav verdi, mens ulike for høyere. Verdier i hjørnene får også lavere verdi fordi dem har færre naboer.

Vi sjekker så om det går mot tap og om det er tilfellet setter vi heuristikken til =tilstandens poengsum*-2. Dette gjør at gjennomsnittsverdien til et bord etter en valgt bevegelse vil få en mye mindre verdi enn en bevegelse som ikke taper. Om det ikke skjer så summerer alle summene fra de tidligere sjekkede og returnerer det som tilstandens heuristikk.

Dette gir følgende heuristikk:

$$VektMatrise = \sum(VektetMatrise * BordMatrise) * 2.2$$

Om høyste verdi er i et hjørne: $VektMatrise = VektMatrise * 1.7$

$$TommeRuter = antallTommeRuter * 20 * \sqrt{}(VektetMatrise * BordMatrise)$$

$$SpredtSum = SpredtSum * 1.7$$

$$Heuristikkpoeng = \sum(VektMatrise + TommeRuter + SpredtSum)$$

Alle konstantene er funnet ved hjelp av nøye testing av mange forskjellige verdier.