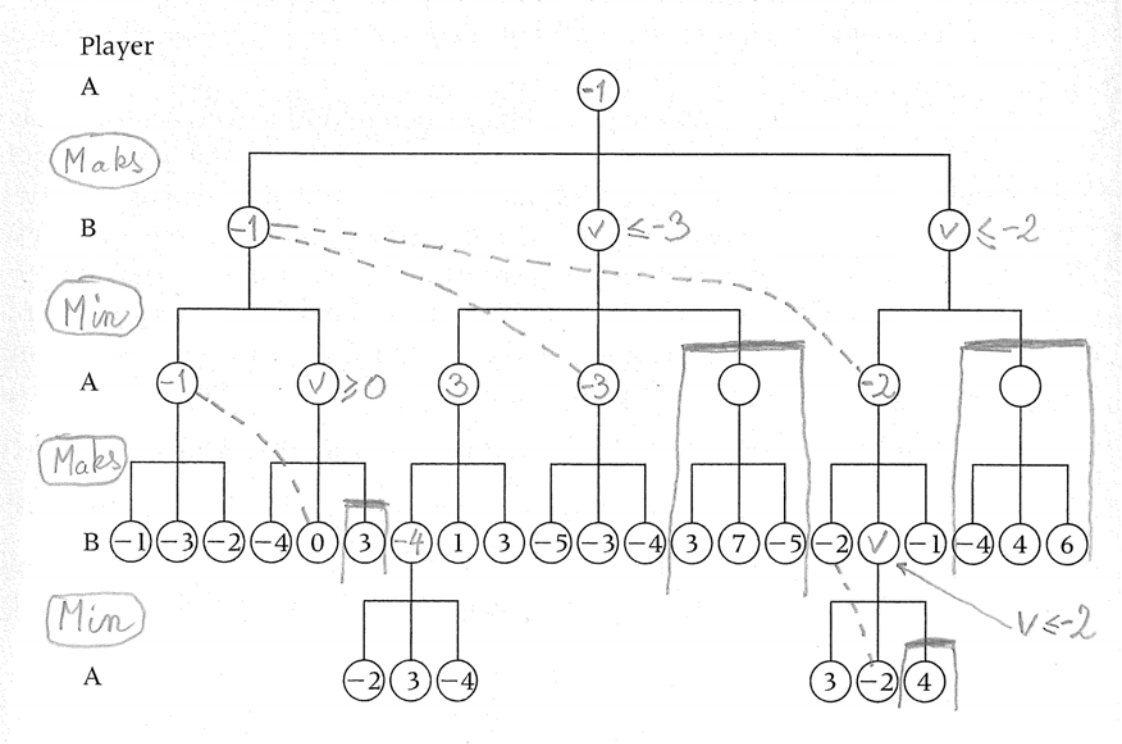
Svarforslag, ukeoppgaver INF 4130, 11. oktober 2011

Oppgave A:
Se på figurene 23.13 og 23.14 (side 735 og 736), der man bruker -1 og +1 for å angi tap/vinst. Gå gjennom treet og sjekk at du er sikker på hvordan de indre nodene får sine verdier ut fra min/max-algoritmen. Pass på å holde fast på at det alltid er vellykketheten i forhold til den som har utgangs-trekket A. For B er det altså bedre jo mindre tallene er. (Men merk altså at i oppgave C er det annerledes. Der negérer vi tallene for hvert nivåskifte, slik at vi hele tiden kan tenke maksimering!).

Svar:
Overlates til gruppene

Oppgave B:
Studer figur 23.16 og 23.17 (side 738 og 740), og sjekk at forståelsen av alfa-beta-avskjæring er på plass. Se etterpå på oppgave 23.22 i boka (side748).

Svar:
Under er figuren fylt ut forhåpentligvis riktig. Alfa- og beta-verdiene har jeg ikke satt inne i nodene, men i stedet angitt det vi vet om den egentlige node-verdien v, slik at det er lett å se at man avskjære. Jeg har også stiplet en linje mellom de to verdiene som er avgjørende for at vi kan avskjære.



Oppgave C:
Gå gjennom programmet på side 741 og diskuter den løsningen som der er valgt med å negére tallene mellom hvert nivå. Merk at det er noen trykkfeil i programmet, se de siste foilene fra 7/10 om alfa-beta-avskjæring. Anta til slutt at nodene (bl.a. X) er objekter av en klasse med attributter "besteTrekk" (typet med denne klassen) og "verdi" (real) som angir det beste trekket fra og alfa/beta-verdien til X sett fra den som er i trekket i situasjonen X.

Svar:
Første del overlates til studier på gruppa. Angående andre del er dette egentlig litt galt stilt, siden vi i en alfa-beta-avskjæring ikke vil finne den beste veien å gå for alle nodene. Analysen gjøres jo på vegne av det som for tiden er rotnoden i treet, og om resultatet for en node ikke lenger har interesse i dens analyse kuttes beregningene. Variablen burde hett "besteTrekkSett", og det er den vi setter inn i programmet under. Det er da viktig at sett fra roten så blir det virkelig det beste trekket (for den som er i trekket) vi får om vi følger "beste trekk sett". Det kommer av at når vi stopper gjennomgangen av barna til X er det nettopp fordi vi innså at foreldren til X ikke ville velge trekket til X som sitt beste. Når det gjelder alfa/beta-verdien er den rett fram å sette riktig.

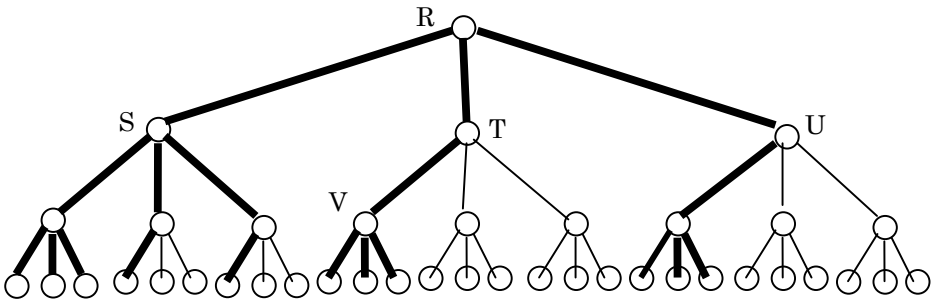
```
1  real function ABNodeValue(
2      X, // Noden vi vil ha alfa/beta-verdien for. Barn: C[1], C[2], … , C[k]
3      numLev, // Antall nivåer som står igjen
4      parentVal) // Alfa/beta-verdien fra forelder-noden (-LB fra foreldren)
5  // returverdi: Den endelige alfa/beta-verdien for noden X
6  {   real LB; // Løpende LowerBound for alfa/beta-verdi for denne noden
7      real ForrigeLB;
8      if <X er terminal-situasjon> then {return <verdien av X for den som har X-trekket>;}
9      else if numLev = 0 then {return <estimat av verdien av X for den som har X-trekket>;}
10     else {
11         LB := - ABNodeValue(C[1], NumLev-1, ∞);
12         X.bestTrekkSett := C[1]; X.verdi = LB;                          // NY!
13         for i := 2 to k do {
14             if LB >= parentValue then { return LB;}
15             else { forrigeLB := LB; LB := max(LB, -ABNodeVal(C[i], Numlev-1, -LB) );
16                    if forrigeLB < LB then {X.besteTrekkSett := C[i]; X.verdi := LB;}  // NY!
17             }
18        }
19     }
20     return LB;
21 }
```

Startkall: situasjonsKvalitet := ABNodeValue(rotnoden, 10, - ∞)

Oppgave D:
I Rune Djurhuus sine sjakk-foiler fra 7/10 står det at om man er så heldig å hele tiden se på det beste trekket først (for den som er i trekket), så får vi meget god avskjæring. Han påstår sågar at om vi går til en dybde d og har forgreningsfaktor b, så blir da søketiden med alfa-beta-avskjæring O(b*1*b*1*b*…), med d faktorer i alt, i stedet for O(b*b*b*b*b*…) som man får uten avskjæring. Vi skal ikke forsøke å vise det, men i stedet se på et konkret tilfelle. Vi setter d=3 og b=3, og får treet angitt under. Angi på treet de grenene du faktisk må gå ned (og derved hvilke du slipper å se på) i treet. Treet har altså 39 kanter. Hvor mange slipper du å se på?

Svar:
Merk altså her at det vi forsøker er å hele tiden se først på det beste trekket for den som er i trekket. Vi **vet** selvfølgelig ikke hvilket det er (da ble jo analysen veldig lett!), men vi må tenke at vi har en heuristikk som er flink til ofte å gi oss beste trekket, ut fra situasjonen i en node. Vi må altså fremdeles gjennomføre den fulle algoritme. Det vi studerer her er altså hvordan algoritmen vil utvikle seg om vi er maksimalt heldig med denne heuristikken i forhold til den gitte startsituasjon.

Under er treet angitt med tykke kanter der algoritmen må gå ned. Som eksempel kan vi se på nodene S, T og U. Vi forutsetter altså at A har største verdien (altså den største verdien av nodene S, T og U, siden vi skal maksimere i roten R). Vi forutsetter også at trekket til V er det beste som kan gjøres fra stillingen T. Det vil si at verdien som kommer tilbake fra V til T allerede er så liten at vi kan se at T-verdien må bli så liten (husk, vi skal minimere i T) at den ikke kan konkurrere med S-verdien når det skal maksimeres i R. Derved behøver vi ikke se på de to gjenværende forgreningene fra T. For U blir det helt tilsvarende.



Vi ser altså at av de 39 kantene så slipper vi med å se på 19, og antall bladnoder vi ser på er 11 av 27. Det siste tallet er viktig fordi vi ved hver bladnode antakelig må gjøre en litt kraftig vurdering av stillingen for komme opp med et tall for stillingens kvalitet (ut fra en passelig heuristikk).

Ekstra til oppgave D: Anta at du er så uheldig å alltid ta det beste trekket *sist*. Vurder om du da kan få noe avskjæring i det hele tatt.

Svar: Man *kan* da få alfa-beta-avskjæring f.eks. om den nest beste kommer først, og det er minst tre barn.

Oppgave E (ikke viktig for kurset):
Anta at du spiller fyrstikkspillet NIM med *to* bunker, at det er *du* som skal trekke, at bunkene *ikke* har like mange fyrstikker, og at ingen bunke er tom. Prøv å finne en enkel strategi slik at du da helt sikkert vinner.

Svar: Ideen er å sørge for at motspilleren stadig kommer i en situasjon der begge bunkene er like store, altså at *du* hele tiden trekker slik at begge bunkene blir like store. Dette holder du på med så lenge den minste av bunkene har minst to fyrstikker. Dermed vil det alltid være like mange og minst to fyrstikker i bunkene når motstanderen skal trekke. Om han/hun trekker slik at den bunken har minst 2 igjen så går det en runde til, ellers vil det være minst 2 fyrstikker i den ene og 1 eller 0 i den andre. Om det er 1 tar du hele den store bunken og om det er 0 tar du slik at det er igjen 1 fyrstikk. I begge disse tilfellene må motstanderen trekke den siste fyrstikken, og tape.