Øvelse nr. 1

Oppgave 1

a) La f være andelen av operasjoner i en beregning som må utføres sekvensielt, $0 \leq f \leq 1$. Vis at speedup med p prosessorer er begrenset av

$$S \leq \frac{1}{f + (1-f)/p} \ .$$

Dette kalles *Amdahls lov*. Anta at $f = 0.1$. Regn ut maksimal speedup for 3, 5, 10 og 100 prosessorer. Hva skjer med effektiviteten hvis $p \to \infty$?

b) Hvor stor kan f maksimalt være dersom vi ønsker 50% effektivitet på MasPar (16384 prosessorer).

c) I praksis ønsker vi ofte å løse større problemer når vi har flere prosessorer. Vanligvis er f en avtakende funksjon av problemstørrelsen. Under disse forutsetningene må Amdahls lov modifiseres. En enkel modell for denne situasjonen er følgende:
Gitt en problemklasse der n angir størrelsen på problemet. Anta at beregningene består av to deler:

- En sekvensiell del som tar t_s tid uansett n.
- En parallelliserbar del som tar $t_p(n)$ tid på én prosessor, $t_p(n)$ er en økende funksjon i n.

Vi har en parallell maskin med p prosessorer og ønsker at beregningene skal utføres i løpet av en fast totaltid T. I løpet av denne tiden skal vi løse et så stort problem som mulig. Vis at speedup er begrenset av

$$S \leq p(1 - t_s/T) + t_s/T \ .$$

Hva skjer med den øvre grensen for effektiviteten hvis vi lar $p \to \infty$?

Oppgave 2

Vi skal summere $n = 2^k$ tall på en PRAM (Parallel Random Access Machine).

a) La $p = n/2$ være antall prosessorer. Vi vet at tidskompleksiteten for summasjon av tallene er gitt som $T_p(n) = k$, hvis vi benytter log–sum algoritmen beskrevet i forelesningen og i boka side 8. Hva blir kostnaden til denne algoritmen? Er den kost optimal?

b) La $p = 2^r$ der $0 \leq r < k$. Hva blir tidskompleksiteten for log–sum algoritmen med så mange prosessorer? Er dette kost optimalt dersom f.eks. $p(n) \leq \sqrt{n}$? Hva om $p(n) \leq n^x$ for $x < 1$?

Oppgave 3

Vi skal finne alle delsummer (prefix sums) av $n = 2^k$ tall $[a_1, a_2, \ldots, a_n]$, dvs.:

$$\sum_{i=1}^{s} a_i \quad \text{for all } 1 \leq s \leq n.$$

Vi antar i hele oppgaven at vi har en PRAM med $p = n$ prosessorer.

a) Først forsøker vi å lage en parallell algoritme ved en 'divide and conquer' (splitt og hersk) metode. La **delsum** være en rutine som returnerer delsummene av n tall;

$$[b_1, b_2, \ldots, b_n] = \textbf{delsum}([a_1, a_2, \ldots, a_n])$$

slik at

$$b_r = \sum_{i=1}^{r} a_i \ .$$

Vis at vi kan finne **delsum**$([a_1, \ldots, a_{2n}])$ fra **delsum**$([a_1, \ldots, a_n])$ og **delsum**$([a_{n+1}, \ldots, a_{2n}])$. Lag en parallel algoritme på denne måten. Tegn opp DAG'en for denne algoritmen. Hvordan egner den seg for en n-prosessor CREW PRAM og for en n-prosessor EREW-PRAM?

b) Vi vil lage en variant som egner seg bedre for en EREW PRAM. Splitt dataene som

$$[a_1, a_3, a_5, \ldots, a_{2n-1}] \ \text{ og } [a_2, a_4, \ldots, a_{2n}] \ ,$$

og benytt **delsum** på hver av disse delene. Det er også mulig å løse denne oppgaven på andre måter, noen er beskrevet i boka, kap. 2.1.