1 Der beste bekannte Algorithmus für ein gegebenes Problem ist in $O(n^2)$.
Bewerten Sie, ob die folgenden Aussagen wahr oder falsch sind.

True	False	
X	○	Ein paralleler Algorithmus mit Laufzeit in $O(\frac{n^2}{p} + n\sqrt{n})$ ist mit $p \leq \sqrt{n}$ Prozessoren kostenoptimal.
X	○	Ein paralleler Algorithmus mit Laufzeit in $O(\frac{n^2}{p} + \log n)$ ist mit p Prozessoren (bis zu einer gewissen Obergrenze) arbeitsoptimal.
X	○	Ein paralleler Algorithmus mit Laufzeit in $O(\frac{n^2}{p} + \log^2 n)$ ist mit p Prozessoren (zu einer gewissen Obergrenze) kostenoptimal.
X	○	Ein paralleler Algorithmus mit Laufzeit in $O(\frac{n^2}{p} + n)$ ist mit p Prozessoren (bis zu einer gewissen Obergrenze) arbeitsoptimal.

$T_{seq} = O(n^2)$

$p = \sqrt{n}$

$T_{par} = \left(\frac{n^2}{p} \cdot n\sqrt{n}\right) \cdot p = n^2 + pn\sqrt{n} = n^2 + n^2 = 2n^2 = O(n^2)$ $\quad T_{par} \in T_{seq}$ ✓

$W_{par} = p \cdot T_{par} = p \cdot \frac{n^2}{p} + \log(n) = n^2 + \log(n) = O(n^2)$ $\quad W_{par} \in T_{seq} \Leftrightarrow W_{par} \in O(n^2)$ ✓

$p = 1: T_{par} = n^2 + \log^2 n = O(n^2)$ $\quad T_{seq} > T_{par} \cdot p \Rightarrow O(T_{par} \cdot p) \in O(T_{seq})$ ✓

$W_{par} = p \cdot T_{par} = p \cdot \frac{n^2}{p} + n = n^2 + n = O(n^2)$ $\quad W_{par} \in T_{seq} \Leftrightarrow W_{par} \in O(n^2)$ ✓

The product C(n) = pTpar(p,n) is the cost of the parallel algorithm:
Total time in which the p processors are occupied

Definition:
Parallel algorithm is called cost-optimal if C(n) = O(Tseq(n)). A cost-optimal algorithm has linear (perhaps perfect) speed-up

Wpar(p,n) = ∑Ti(n) is the parallel work of the parallel algorithm: total number of instructions performed by p processors

Definition:
Parallel algorithm is called work-optimal if Wpar(p,n) = O(Tseq(n)). A work-optimal algorithm has potential for linear speed-up (for some number of processors)

Frage 3
Bisher nicht beantwortet
Erreichbare Punkte: 4,00
Frage markieren

Bewerten Sie, ob die folgenden Aussagen zur parallelen Effizienz von parallelen Algorithmen wahr oder falsch sind.

True	False	
○	X	Die parallele Effizienz eines Algorithmus wird immer in Relation zur Laufzeit desselben Algorithmus berechnet, wenn dieser mit nur einem Prozessor ausgeführt wird.
X	○	Die parallele Effizienz gibt an, wie nah die Laufzeit $T_{Par}(p,n)$ an die best-mögliche Parallelisierung herankommt, also $\frac{T_{seq}(n)/p}{T_{par}(p,n)}$.
○	X	Die parallele Effizienz kann größer als 1 sein (man spricht dann von "Super-Effizienz").
X	○	Die parallele Effizienz wird berechnet indem die sequenzielle Laufzeit (eines best-bekannten/-möglichen Algorithmus) durch die Kosten des parallelen Algorithmus mit p Prozessoren geteilt wird.

$\Rightarrow \frac{T_{par}(n,p)}{T_{par}(n,1)}$

Definition:
The efficiency of parallel algorithm Par is the ratio of best possible parallel time to actual parallel time for given p and n:

E(p,n) = (Tseq(n)/p) / Tpar(p,n)
= S_p(n)/p = Tseq(n) / (p Tpar(p,n))

Cost, so efficiency is also ratio of sequential to parallel cost

Remarks:

Frage 4
Bisher nicht beantwortet
Erreichbare Punkte: 4,00
Frage markieren

Ein (nicht sonderlich guter) paralleler Sortieralgorithmus läuft mit p Prozessoren in $O(\frac{n \log n}{p} + n)$. Die sequenzielle "Baseline" läuft in $O(n \log n)$ mit vergleichbaren Konstanten (hinter dem O versteckt). Wir erlauben uns diese Konstanten zu ignorieren.

Bewerten Sie, ob die folgenden Aussagen wahr oder falsch sind.

True	False	
X	○	Der parallele Algorithmus ist schwach skalierend, wenn eine konstante parallele Effizienz beibehalten werden soll.
X	○	Die Isoeffizienz-Funktion kann nicht analytisch berechnet werden - zumindest finde ich kein geschlossen Ausdruck ("closed form") mit den mir bekannten Methoden; sowie im Beispiel auf den Folien.
X	○	Die Isoeffizienz-Funktion ist $n = \exp(\frac{e}{1-e}p)$ (Zwischenschritt $\log n = \frac{e}{1-e}p$), wobei e die parallele Effizienz ist, die beibehalten werden soll.
○	X	Die Isoeffizienz-Funktion ist $p = \log n \frac{1-e}{e}$, wobei e die parallele Effizienz ist, die beibehalten werden soll.

Definition:
A parallel algorithm/implementation is weakly scaling if there is a slowly growing function f(p), such that for n = Ω(f(p)), E(p,n) remains constant. The function f is called the iso-efficiency function

If the algorithm instead runs in $O(n^2/p+\log^2 n)$, the iso-efficiency function for this algorithm ("how must problem size n increase as a function of p to maintain constant efficiency?") is

$e = n^2/(p(n^2/p+\log^2 n)) = n^2/(n^2+p\log^2 n) \Leftrightarrow$
$n^2(1-e) = e\,p\log^2 n \Leftrightarrow$
$n/\log n = \sqrt{(e/(1-e))}\sqrt{p}$

O-constants normalized to 1

No analytical solution

$e = \frac{n \cdot \log n}{(\frac{n \cdot \log n}{p} + n) \cdot p}$

$e = \frac{n \cdot \log n}{n \cdot \log n + n \cdot p}$

$e \cdot n \cdot \log n + e \cdot n \cdot p = n \cdot \log n$

$e \cdot n \cdot p = n \cdot \log n (1-e)$

$e \cdot p = \log n (1-e)$

$\frac{e \cdot p}{1-e} = \log n$

$10^{\frac{e \cdot p}{1-e}} = n$

Frage **5**

Bisher nicht beantwortet

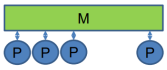
Erreichbare Punkte: 4,00

Frage markieren

Bewerten Sie, ob die folgenden Aussagen zum PRAM-Modell wahr oder falsch sind.

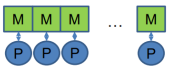
True	False	
X	○	Das PRAM-Modell ist ein UMA-Modell.
○	X	Das PRAM-Modell ist ein NUMA-Modell.
X	○	Das PRAM-Modell ist ein MIMD-Modell.
○	X	Das PRAM-Modell ist ein MISD-Modell.

UMA (Uniform Memory Access):
Access time to memory location independent of location and accessing processor, e.g., O(1), O(log M), …



Examples: RAM is UMA, PRAM is UMA (unit cost)

NUMA (Non-Uniform Memory Access):
Access time depends on processor and location.



(Almost) All "real" processors are NUMA

Frage **6**

Bisher nicht beantwortet

Erreichbare Punkte: 4,00

Frage markieren

Ein paralleler (PRAM) Algorithmus hat die unten beschriebenen Eigenschaften.

Bewerten Sie, ob die folgenden Aussagen wahr oder falsch sind.

True	False	
○	X	Eigenschaft: Der Algorithmus arbeitet in (synchronisierten) Runden. Die Problemgröße wird in jeder Runde mit einem konstanten Faktor k kleiner. Der Algorithmus terminiert, wenn sich die Problemgröße auf $n = 1$ verkleinert hat. Aussage: Der Algorithmus braucht $\log_{k+1} n$ Runden (Logarithmus zur Basis $k + 1$).
X	○	Eigenschaft: Die parallele Arbeit in jeder Runde des Algorithmus liegt in $O(n)$, wobei n die Problemgröße für die jeweilige Runde ist. Die Problemgröße n wird in jeder Runde halbiert und der Algorithmus terminiert, wenn sich die Problemgröße auf $n = 1$ verkleinert hat. Aussage: Die Gesamtarbeit aller Runden ist in $O(N \log N)$, wobei N die Problemgröße der Eingabe ist (also $n = N$ bevor die erste Runde anfängt).
○	X	Eigenschaft: Die parallele Arbeit in jeder Runde des Algorithmus liegt in $O(n)$, wobei n die Problemgröße für die jeweilige Runde ist. Die Problemgröße n wird in jeder Runde halbiert, und der Algorithmus terminiert, wenn sich die Problemgröße auf $n = 1$ verkleinert hat. Aussage: Die Gesamtarbeit aller Runden ist in $O(N)$ (linear), wobei N die Problemgröße der Eingabe ist (also $n = N$ bevor die erste Runde anfängt).
X	○	Eigenschaft: Der Algorithmus arbeitet in (synchronisierten) Runden. Die Problemgröße n wird in jeder Runde halbiert. Der Algorithmus terminiert, wenn sich die Problemgröße auf $n = 1$ verkleinert hat. Aussage: Der Algorithmus braucht daher $\log_2 n$ Runden.

runden = $\left\lceil \frac{n}{k} \right\rceil$

Frage **7**

Bisher nicht beantwortet

Erreichbare Punkte: 4,00

Frage markieren

Es sei folgender paralleler PRAM-Algorithmus in Pseudocode gegeben:

```
nn = n;
while (nn>1) {
  k = nn/2+nn%2;
  par (0<=i<k) {
    if (i+k<nn) a[i] = a[i]+a[i+k];
  }
  nn = k;
}
s = a[0];
```

Bewerten Sie, ob die folgenden Aussagen wahr oder falsch sind.

True	False	
X	○	Der Algorithmus berechnet die Summe $s = a[0] + a[1] + \ldots + a[n-1]$.
X	○	Der Algorithmus terminiert in $O(\log n)$ Zeitschritten.
X	○	Die Arbeit (work) des Algorithmus ist in $O(n^2)$.
○	X	Der Algorithmus würde auch mit einer nicht-kommutativen, binären Operation $\odot$ statt "+" funktionieren ($a[0] \odot a[1] \neq a[1] \odot a[0]$).

Frage **8**
Bisher nicht beantwortet
Erreichbare Punkte: 4,00
Frage markieren

Bewerten Sie, ob die folgenden Aussagen zu Präfixsummen-Algorithmen wahr oder falsch sind.

True	False	
X	○	Das "Inclusive Prefix-Sums"-Problem mit einem Feld der Länge n kann auf einer EREW PRAM in $O(\frac{n}{p} + \log p)$ Zeitschritten gelöst werden.
X	○	Es existieren arbeitsoptimale Algorithmen für das (inklusive und exklusive) Präfixsummen-Problem.
X	○	Das "Exclusive Prefix-Sums"-Problem mit einem Feld der Länge n kann auf einer EREW PRAM in $O(\frac{n}{p} + \log n)$ Zeitschritten gelöst werden.
X	○	Der Speed-up für einen arbeitsoptimalen Präfixsummen-Algorithmus kann in $O(p)$ sein, wenn p in $O(\frac{n}{\log n})$ ist.

With the blocking technique explained next, the result can be improved.

Theorem:
The (inclusive/exclusive) prefix-sums problem for an array of n elements with an associative binary operator "+" can be solved on an EREW PRAM with p processors in O(n/p+log p) time steps.

Three theoretical solutions to the parallel prefix-sums problem

1. Recursive: Fast, work-optimal
2. Iterative: Fast, work-optimal
3. Doubling: Fast(er), not work-optimal (but still useful)

Prefix-sums on the PRAM

With some care, both recursive and non-recursive prefix-sums algorithms for the inclusive-prefix-sums problem can be implemented on the PRAM

Home exercise

Theorem:
The (inclusive/exclusive) prefix-sums problem for an array of n elements with an associative binary operator "+" can be solved on an EREW PRAM with p processors in O(n/p+log n) time steps.

1. Recursive solution: Summary

- With enough processors, T∞(n) = 2 log n parallel steps (recursive calls) needed, two barrier synchronizations per recursive call
- Number of operations, W(n) = O(n), all perfectly parallelizable (data parallel)
- Tpar(p,n) = W(n)/p + T∞(n) = O(n/p+log n)
- Linear speed-up up to Tseq(n)/T∞(n) = n/log n processors

$T_{seq} = O(n)$

$T_{par} = O(\frac{n}{p} + \log n)$

$\frac{T_{seq}}{T_{par}} = \frac{n}{\frac{n}{\log n} + \log n} = \frac{n}{\frac{\log n \cdot n}{n} + \log n} = \frac{n}{2 \log n}$

Frage **9**
Bisher nicht beantwortet
Erreichbare Punkte: 4,00
Frage markieren

Bewerten Sie, ob die folgenden Aussagen zum Speed-up wahr oder falsch sind.

True	False	
○	X	Unter der Annahme, dass ein Speed-up nie "superlinear" ist, ist der absolute Speed-up immer größer als der relative Speed-up.
○	X	Fällt der Speed-up bei einer bestimmten Anzahl von Prozessoren, dann steigt die parallele Effizienz.
○	X	Unter "perfektem Speed-up" verstehen wir einen Speed-up in $O(1)$.
X	○	Der relative Speed-up gibt Aufschluss über die Skalierbarkeit eines parallelen Algorithmus, wenn dieser Algorithmus mit p Prozessoren zu sich selber mit einem Prozessor verglichen wird.

$\text{relativ} = \frac{T_{par}(p,7)}{T_{par}(p,n)} = \frac{T_{par}(p,7)}{T_{par}(p,7)} = 1$ $\quad \text{absolut} = \frac{T_{seq}}{T_{par}} \geq 1$

$E(p,n) = \frac{S(n)}{p} \approx$ fällt ab

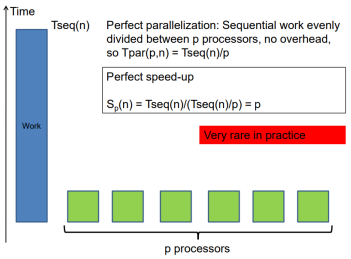
p	5	6	7
S	10	10	7

$O(p)$

Definition:
The efficiency of parallel algorithm Par is the ratio of best possible parallel time to actual parallel time for given p and n:

E(p,n) = (Tseq(n)/p) / Tpar(p,n)
= S_p(n)/p = Tseq(n) / (p Tpar(p,n))

Cost, so efficiency is also ratio of sequential to parallel cost



Frage **10**
Bisher nicht beantwortet
Erreichbare Punkte: 4,00
Frage markieren

Im Folgenden seien einige endliche Summen gegeben.

Bewerten Sie, ob die folgenden Aussagen wahr oder falsch sind.

True	False	
X	○	Die Summe von $1 + r + r^2 + \ldots + r^k$ ist $\frac{r^{k+1}-1}{r-1} < \frac{1}{1-r}$, wenn $\lvert r \rvert < 1$.
○	X	Die Summe $1 + 2 + 4 + 8 + 16 + \ldots + 2^k$ ist 2^{k+1}.
X	○	Die Summe $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots + \frac{1}{2^k}$ ist kleiner als 1.
X	○	Die Summe von $1 + q + q^2 + \ldots + q^k$ ist, wenn $q \neq 1$, $\frac{1-q^{k+1}}{1-q}$.

Frage **2**
Bisher nicht beantwortet
Erreichbare Punkte: 4,00
Frage markieren

Ein paralleles Programm hat von außen betrachtet Eigenschaften, die vom Amdahlschen Gesetz beschrieben werden können.

Bewerten Sie, ob die folgenden Aussagen wahr oder falsch sind.

True	False	
○	X	Es sei p die Anzahl der Prozessoren, f der perfekt parallelisierbare Anteil des Programms und n die Eingabegröße. Der größtmögliche Speed-up, den das Programm erreichten kann, ist nur dann begrenzt durch $\frac{1}{1-f}$, wenn $p \leq fn$.
○	X	Für eine Eingabe I braucht das Programm 1000 Sekunden auf einem Rechnersystem mit einem Prozessor. Hiervon werden immer 25 Sekunden strikt sequenziell ausgeführt. Der maximale Speed-up für Eingabe I mit einer beliebig großen Anzahl von Prozessoren beträgt dann 10.
X	○	Es sei p die Anzahl der Prozessoren, f der perfekt parallelisierbare Anteil des Programms und n die Eingabegröße. Der größtmögliche Speed-up, den das Programm erreichten kann, ist begrenzt durch $\frac{1}{1-f}$.
○	X	Die Speed-up-Limitierung, die in Amdahls Gesetz beschrieben ist, kann immer umgangen werden indem die Eingabegröße entsprechend erhöht wird.

$$S = \frac{T_{seq}}{T_{par}} = \frac{1000}{25 + \frac{975}{p}} = \lim_{p \to \infty} \frac{1000}{25 + \frac{975}{p}} = \frac{1000}{25} = 40$$