Steven Müller 581583
Lucas Petersen 581277

Aufgabe 3

a) z.z. $count(p) \geq count(u)$. wobei u ein Kind von p ist.

Zwei Fälle:
1. u ist ein Blatt:

$$count(p) \geq 1$$

Da u ein Kind von p mindestens eine Instanz von p muss es existieren, somit es gilt.

2. u ist ein innere Knoten:

$$\lceil \frac{\sum_{i=1}^{l} size(p_i)}{r(p) - m(p)} \rceil \geq \lceil \frac{\sum_{j=1}^{k} size(u_j)}{r(u) - m(u)} \rceil$$

Wir minimieren $\sum_{i=1}^{l} size(v_i)$ indem wir sagen, dass u das einzige Kind von p ist.

$$\lceil \frac{size(u)}{r(p) - m(p)} \rceil \geq \lceil \frac{\sum_{j=1}^{k} size(u_j)}{r(u) - m(u)} \rceil$$

$$\lceil \frac{\sum_{j=1}^{k} size(u_j) + m(u) \cdot count(u)}{r(p) - m(p)} \rceil \geq \lceil \frac{\sum_{j=1}^{k} size(u_j)}{r(u) - m(u)} \rceil$$

$$\lceil \frac{\sum_{j=1}^{k} size(u_j) + m(u) \cdot count(u)}{M - \sum_{p' \in anc(p)} m(p') - m(p)} \rceil \geq \lceil \frac{\sum_{j=1}^{k} size(u_j)}{M - \sum_{u' \in anc(u)} m(u') - m(u)} \rceil$$

Gilt wenn:

$$(1) \sum_{j=1}^{k} size(u_j) + m(u) \cdot count(u) \geq \sum_{j=1}^{k} size(u_j)$$

und (aus (1)):

$$(2) M - \sum_{v' \in anc(p)} m(p') - m(p) \leq M - \sum_{u' \in anc(u)} m(u') - m(u) + m(u) \cdot count(u)$$

Zu (1):

$$\sum_{j=1}^{k} size(u_j) + m(u) \cdot count(u) \geq \sum_{j=1}^{k} size(u_j)$$

Gilt, da $count(u) \geq 1$ und $m(u) \geq 0$.

Zu (2):

$$M - \sum_{p' \in anc(p)} m(p') - m(p) \leq M - \sum_{u' \in anc(u)} m(u') - m(u) + m(u) \cdot count(u)$$

$$- \sum_{p' \in anc(p)} m(p') - m(p) \leq - \sum_{u' \in anc(u)} m(u') - m(u) + m(u) \cdot count(u)$$

$$\sum_{p' \in anc(p)} m(p') + m(p) \geq \sum_{u' \in anc(u)} m(u') + m(u) - m(u) \cdot count(u)$$

Da u das einzige Kind von p ist haben sie die gleiche Blätter somit $m(u) = m(p)$ und:

$$\sum_{p' \in anc(p)} m(p') + m(p) \geq \sum_{u' \in anc(u)} m(u') + m(p) - m(u) \cdot count(u)$$

$$\sum_{p' \in anc(p)} m(p') \geq \sum_{u' \in anc(u)} m(u') - m(u) \cdot count(u)$$

Da $anc(p) \cup p = anc(u)$ und $m(u) = m(p)$:

$$\sum_{p' \in anc(p)} m(p') \geq \sum_{u' \in anc(p)} m(p') + m(u) - m(u) \cdot count(u)$$

$$0 \geq m(u) - m(u) \cdot count(u)$$

$$m(u) \cdot count(u) \geq m(u)$$

$$count(u) \geq 1$$

Gilt. Somit ist bewiesen dass $count(p) \geq count(u)$.
b)
Erstens für count(v) ist eine untere Schranke für die Anzahl von Instanzen:
Zwei Fälle:

1. v ist ein Blatt:

$$count(v) = 1$$

Gilt: da wir nicht weniger als 1 Kopie haben können.
2. v ist kein Blatt:

$$count(v) = \lceil \frac{\sum_{i=1}^{l} size(v_i)}{r(v) - m(v)} \rceil$$

Unter der Annahme, dass size(v) gibt eine untere Schranke für der Speicherverbrauch von v(siehe unten). Es müssen mindestens so viele Kopien von v da sein, sodass alle Kinder von v gespeichert werden können bei der verbleibende Restkapazität. Keine weitere Blätter sind in den Rechner außer die Kinder von v brauchen $count(v) = \lceil \frac{\sum_{i=1}^{l} size(v_i)}{r(v)-m(v)} \rceil$ viele Kopien von v somit ist $count(v)$ eine untere Schranke für die Anzahl von benötige Kopien von v.

Für size(v) als untere Schranke für den Speicherverbrauch:
1. v ist ein Blatt:

$$size(v) = m(v)$$

Gilt da den Speicherverbrauch selbst eine untere Schranke für den Speicherverbrauch ist.
2. v ist kein Blatt:

$$size(v) = \sum_{i=1}^{l} size(v_i) + m(v) \cdot count(v)$$

Alle Kinder von v müssen gespeichert werden und so werden wir mindestens $\sum_{i=1}^{l} size(v_i)$ Platz brauchen. Wir müssen auch der Knoten v selbst speichern und das so oft wie es Kopien davon gibt, somit brauchen wir $\sum_{i=1}^{l} size(v_i) + m(v) \cdot count(v)$.

Aufgabe 4

a) Für unser Preprocessing-Schritt wir berechnend die Wertdichte alle Items (Wert durch Gewicht). Dies läuft in $\mathcal{O}(n)$. Dann wir sortieren alle Item nach Wertdichte mit z.B. Merge-Sort in $\mathcal{O}(n \cdot log(n))$. Letztlich, wir gehen alle Items durch in der sortierte Reihenfolge und speichern der Gewicht der bis alle vorherigen Items plus den aktuellen betrachten Item. Somit wird $w(a_1)$ im Item a_1, $w(a_1) + w(a_2)$ in a_2 und $\sum_{i=1}^{k} w(a_i)$ in Item a_k gespeichert. Da dafür wir einfach durch die Items linear durchgehen brauchen wir nur $\mathcal{O}(n)$. Weiterhin wir tun das gleiche für den Wert der Items, sodass $\sum_{i=1}^{k} v(a_i)$ in Item a_k gespeichert.
Somit braucht der Preprocessing-Schritt insgesamt $\mathcal{O}(n \cdot log(n))$.

Jetzt um ein upperBound zu berechnen wir können einfach eine binäre Suche einsetzen.
Wir geben ein unser maximal Gewicht und suchen binär nach diesem Wert beim Präfixsummen der Gewichten. Falls wir den genauen Gewicht finden geben wir einfach die Präfixsumme der Werte an der Stellen. Sonst wir sollen einfach der nächst kleineren Präsfixsummengewicht(dies geht immer noch in $\mathcal{O}(log(n))$) und dann wir gucken den nächsten Item(a_{k+1}) um folgende Gleichung zu lösen:

$$c_{up} = \frac{(c_{max} - \sum_{i=1}^{k} w(a_i))}{w(a_{k+1})} \cdot v_{k+1} + \sum_{i=1}^{k} v(a_i)$$

Da $\sum_{i=1}^{k} w(a_i)$ und $\sum_{i=1}^{k} v(a_i)$ im a_k gespeichert sind geht dies im $\mathcal{O}(1)$. c_{up} ist der upperBound und c_{max} ist der maximalen Gewicht. Somit wird er upperBound insgesamt in $\mathcal{O}(log(n))$ berechnet.