ADA hw4 – 2

B08902065 資工二 洪易

Cite:

B08902013 張永達 B08902127 林歆凱 B08902063 陳羿穎 B08902075 林耘平

3. Lonely Christmas

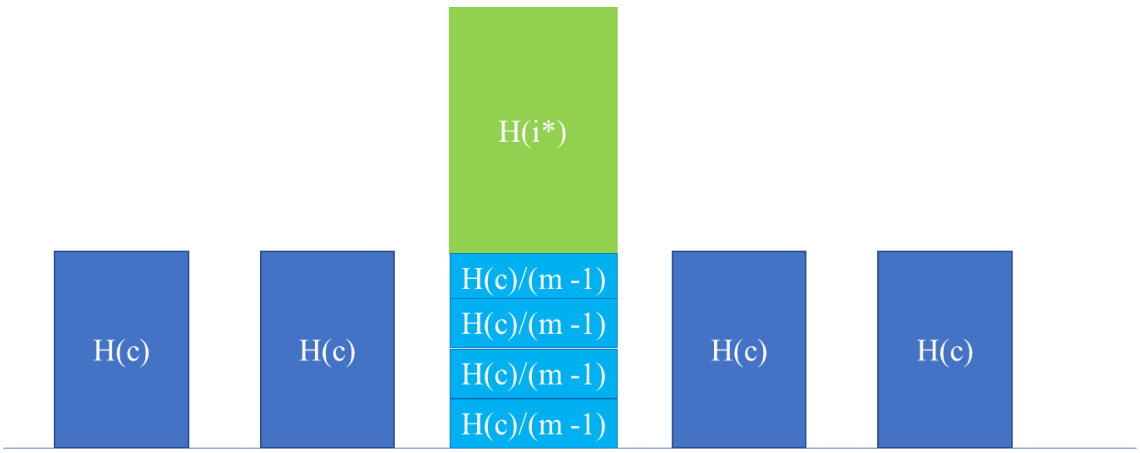
(a)

To TA:

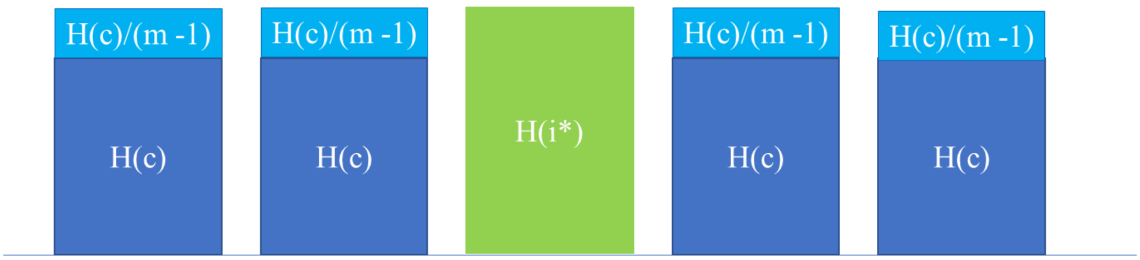
I first think of an interesting solution, but I was convinced by peers that sol_2 is better, I'd like to make sure if sol_1 is also an acceptable solution? Could you please grade sol_2 and also tell me whether sol_1 is OK? Thanks.

Sol_1:

We can see that there will exist i* that decide H_{max}. It's trivial that the worst case (max (H_{max} / OPT(I))) happens when h(i*) is placed while current height of all towers is same and it's placed last. Let's call the current height before i* is placed h(c). Therefore, H_{max} = h(i*) + h(c). However, OPT(I) = max (h(i*), h(c) * m / (m - 1)) as the following figure shown. As a result, while m → ∞ and h(i*) = h(c), we can see that OPT(I) = 2 * H_{max} in the worst case. Otherwise, 2 * H_{max} < OPT(I), and thus it's a *2-approximation* algorithm. The following two graphs show the worst case in the algorithm of H_{max} and OPT(I), I take m = 5 for instance.



H_{max} = h(i*) + h(c)



OPT(I) = max (h(i*), h(c) * m / (m - 1))

When m → ∞, and h(i*) = h(c), the maximum (H_{max} / OPT(I)) occurs, which is 2.

Sol_2:

I split the height of tower i* is into two part, $H_{max} = h(i^*) + (H_{max} - h(i^*))$. It's trivial to say that $h(i^*) \leq \mathrm{OPT}(I)$. Moreover, $(H_{max} - h(i^*)) \leq \frac{\text{sum\_height\_of\_boxes}}{number\_of\_boxes} \leq$ OPT(I), Thus $H_{max} \leq 2 \times \mathrm{OPT}(I)$. It's a *2-approximation* algorithm.

(b)

I consider two cases. I* in B' and i* not in B'.

First, if I* in B', then it's arranged by brute force algorithm and be placed in optimal way. Therefore, the H_{max} = OPT(I) in this case.

Otherwise, if I* is not in B'. I split the height of tower i* is into two part, $H_{max} = h(i^*) + (H_{max} - h(i^*))$. It's trivial to say that $h(i^*) \leq \epsilon * \mathrm{OPT}(I)$. Moreover, $(H_{max} - h(i^*)) \leq \frac{\text{sum\_height\_of\_boxes}}{number\_of\_boxes} \leq$ OPT(I), Thus $H_{max} \leq (1+\epsilon) \times \mathrm{OPT}(I)$.

In both cases, $H_{max} \leq (1+\epsilon) \times \mathrm{OPT}(I)$, it's a $1+\epsilon$-*approximation* algorithm.

(c)

Same as (b). Let's consider two cases, i* in B and i* not in B.

First, if I* in B', which means that ∀ h'(i) $= \left\lfloor \frac{h(i)}{\mu} \right\rfloor \times \mu > \left\lfloor \frac{\epsilon v}{\epsilon^2 v} \right\rfloor \times \epsilon^2 V = \left\lfloor \frac{1}{\epsilon} \right\rfloor \times \epsilon^2 V = \epsilon V$. We can see that h'(i) > ϵV, and $H_{max} \leq V$, thus, for each tower, the maximum boxes there can be placed is at most $\frac{1}{\epsilon}$. Moreover, for each box, the maximum error between h(i) and h'(i) is $|\, h(i) - h'(i) \,| < 1 * \mu = \epsilon^2 V$. Therefore, the maximum error between H_{max} come from h' and H_{max} come from h is $\frac{1}{\epsilon} \times \epsilon^2 V = \epsilon V$. Thus, as H_{max} from h' ≤ V, H_{max} from h ≤ V + ϵV = (1 + ϵ) × V .

Second, if I* not in B'. We split the height of tower i* is into two part, H_{max} = h(i*) + (H_{max} - h(i*)). It's trivial to say that h(i*) ≤ ϵV. Moreover, (H_{max} – h(i*)) ≤ $\frac{\text{all\_sum\_of\_boxes}}{number\_of\_boxes}$ ≤ OPT(I) ≤ V, else i* won't be placed on this tower. As a result, H_{max} = h(i*) + (H_{max} - h(i*)) ≤ ϵV + V = (1 + ϵ) × V.

According to two cases, H_{max} ≤ (1 + ϵ) × V. QED.

(d - 1)

If h'(i) $\leq$ V. It's trivial that $\mu = \epsilon^2 V$, and $|\vec{n}| \times \mu \leq V$. Thus $|\vec{n}| \leq \frac{V}{\mu} = \frac{1}{\epsilon^2}$.

Else, if h'(i) > V, return false, no $\vec{n}$.

Therefore, $|\vec{n}|$ is bounded by $\frac{1}{\epsilon^2}$, QED.

(d - 2)

$$F(\vec{n}) = \begin{cases} 1 & if\ \vec{n}\ in\ U \\ 1 + min\ (F(\vec{n} - \vec{u}), \forall\ \vec{u}\ in\ U) & if\ \vec{n}\ not\ in\ U \end{cases}$$

(d - 3)

Let's split the process into two steps.

First, constructing U. As there are at most $\frac{1}{\epsilon^2}$ boxes in a tower and $\frac{1}{\epsilon^2}$ kind of boxes.

Construct U takes O $(\frac{1}{\epsilon^2}^{\frac{1}{\epsilon^2}})$ = O (1) time.

Second, fill in the dynamic programming table. As we need to calculate all possibility of $\vec{n} - \vec{u}$, it equals to $\prod_{i=1}^{\lfloor\frac{1}{\epsilon^2}\rfloor}(n\_init_i + 1) \leq B^{\lfloor\frac{1}{\epsilon^2}\rfloor} = O(B^{\frac{1}{\epsilon^2}})$, as (n_init$_i$ + 1) $\leq$ B (n_init$_i$ means the i$_{th}$ entry of $\overleftarrow{n\_init}$). Accordingly, the total time complexity of this algorithm is O (1) + $O(B^{\frac{1}{\epsilon^2}}) = O(B^{\frac{1}{\epsilon^2}})$.

(e)

I apply binary search on V and use function Partial_Rounded (I, V) to find a $1 + \epsilon - approximation$ algorithm. Search start from (0, max_V $\leq 2^l$), check what is the maximum V that ORACLE (I', V) won't return false, and then put all box that's not in B into the towers in greedy algorithm. Finally, I find out the answer.

For the time complexity, as max_V $\leq 2^l$ and I use binary search in the algorithm, we can achieve an algorithm of $O(\log_2 V) * O(B^{\frac{1}{\epsilon^2}}) = O(l * B^{\frac{1}{\epsilon^2}})$, which is polynomial.