ShanghaiTech University

CS240 Algorithm Design and Analysis

Fall 2022

Problem Set 4 Answer

Due: 23:59, Dec. 31, 2022

1. Submit your solutions to Gradescope (www.gradescope.com).
2. In "Account Settings" of Gradescope, set your FULL NAME to your Chinese name and enter your STUDENT ID correctly.
3. If you want to submit a handwritten version, scan it clearly. Camscanner is recommended.
4. When submitting your homework, match each of your solution to the corresponding problem number.

Problem 1:

Suppose that there are n items need to be placed in some bins. The capacity of each bin is 1, and the volume of items are not same and all volumes are smaller than 1. We want to use the fewest number of bins to place all items. Please design a 2-approximation algorithm to solve this problem.

Solution:

For each item, puts it in one of partially packed bins if it can be placed into.If all the partially packed bins can not fit the item, then we use a new bin to place the item.The number of the bins is at most 2 times of the optimal solution.

Proof: Let k^* denote the optimal solution. Suppose that the algorithm above returns k bins.Then we can know that at least k-1 bins are more than half full.Hence the optimal solution k^* >k-1 i.e. $2k^*$ >k-1.Since the number of bins is integer, we can get $k \leq 2k^*$ i.e.the algorithm above is 2-approximated to solve this problem.

Problem 2:

In HW3, you've considered the problem that optimally assign m items to two players. Now suppose there are n players and each of them has a unique valuation and an upper bound. Provide a 2-approximation algorithm to solve the problem and prove it.

Solution:

Proof. Let Q be the original problem and define Q' to be the problem on the $m-1$ remaining items after item 1 is removed: i.e., item 1 is unavailable and v_j is replaced by v_j' with $v_j'(S) = v(S \mid \{1\}) = v(S \cup \{1\}) - v(\{1\})$, where j is the player to which item 1 was allocated. All other valuations $v_i, i \neq j$ are unchanged. Notice that the algorithm above may be viewed as first allocating item 1 to j and then allocating the other elements using a recursive call on Q'.

Let us denote by $ALG(Q)$ the value of the allocation produced by this algorithm, and by $OPT(Q)$ the value of the optimal allocation. Let $p = v_j(\{1\})$. By the definition of Q', it is clear that $ALG(Q) = ALG(Q') + p$. We will now show that $OPT(Q) \leqslant OPT(Q') + 2p$. Let $S = (S_1, \ldots, S_n)$ be the optimal allocation for Q, and assume that $1 \in S_k$, i.e., item 1 is allocated to bidder k by the algorithm above. Let $S' = (S_1', \ldots, S_n') = (S_1, \ldots, S_k - \{1\}, \ldots, S_n\ )$ be the solution to Q' obtained from S by ignoring item 1. Let us compare the value of S' in Q' (which is a lower bound on $OPT(Q')$) to the value of S in Q (which is exactly $OPT(Q)$). All players except k get the same allocation and all players except j have the same valuation. Without loss of generality, assume $k \neq j$. Player k loses at most $v_k(\{1\})$. But $v_k(\{1\}) \leqslant v_j(\{1\}) = p$ and player k looses at most p. Player j looses at most p since, by monotonicity of $v_j, v_j'(S_j) = v_j(S_j \cup \{1\}) - v_j(\{1\}) \geqslant v_j(S_j) - p$. Therefore $OPT(Q') \geqslant OPT(Q) - 2p$. The proof is concluded by induction on Q':

$$OPT(Q) \leqslant OPT(Q') + 2p \leqslant 2ALG(Q') + 2p = 2ALG(Q)$$

Problem 3:

Suppose to perform a sequence of n operations on a data structure. The i-th operation costs i if i is an exact power of 2, otherwise i-th operation costs 1.
a) Using accounting method to determine the amortized cost per operation.
b) Using potential method to determine the amortized cost per operation.

Solution:

a) Accounting Method

Let c_i = cost of i th operation.

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2,} \\ 1 & \text{otherwise} \end{cases}$$

Charge 3 (amortized cost $\hat{c}_i$) for each operation.

- If i is not an exact power of 2 , pay \$1, and store \$2 as credit.
- If i is an exact power of 2 , pay \$$i$, using stored credit.

Operation	Cost	Actual cost	Credit remaining
1	3	1	2
2	3	2	3
3	3	1	5
4	3	4	4
5	3	1	6
6	3	1	8
7	3	1	10
8	3	8	5
9	3	1	7
10	3	1	9
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Since the amortized cost is \$3 per operation, $\sum_{i=1}^{n} \hat{c}_i = 3n$. We know from Exercise 17.1-3 that $\sum_{i=1}^{n} \hat{c}_i < 3n$. Then we have

$$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i \Rightarrow \text{ credit } = \text{ amortized cose } - \text{ actual cost } \geq 0.$$

Since the amortized cost of each operation is $O(1)$, and the amount of credit never goes negative, the total cost of n operations is $O(n)$.

b) Potential Method

Define the potential function $\Phi(D_0) = 0$, and $\Phi(D_i) = 2i - 2^{1+\lfloor \lg i \rfloor}$ for $i > 0$. For operation 1 ,

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 2i - 2^{1+\lfloor \lg i \rfloor} - 0 = 1.$$

For operation $i (i > 1)$, if i is not a power of 2 , then

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 1 + 2i - 2^{1+\lfloor \lg 1 \rfloor} - \left(2(i-1) - 2^{1+\lfloor \lg (i-1) \rfloor}\right) = 3.$$

If $i = 2^j$ for some $j \in \mathbb{N}$, then

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = i + 2i - 2^{1+j} - \left(2(i-1) - 2^{1+j-1}\right) = i + 2i - 2i - 2i + 2 + i = 2$$

Thus, the amortized cost is 33 per operation.

Problem 4:

Given a function rand2() that returns 0 or 1 with equal probability, implement rand3() using rand2() that returns 0, 1 or 2 with equal probability. Minimize the number of calls to rand2() method. Prove the correctness.

Solution:

The idea is to use expression 2rand2() + rand2(). It returns 0, 1, 2, 3 with equal probability. To make it return 0, 1, 2 with equal probability, we eliminate the undesired event 3.

$$P(0) = P(1) = P(2) = P(3) = 0.25$$

Thus, 0, 1, 2 are in the same probability.

Problem 5:

Suppose that for some decision problem, we have an algorithm which on any instance computes the correct answer with probability at least 2/3. We wish to reduce the probability of error by running the algorithm n times on the same input using independent randomness between trials and taking the most common result. Using Chernoff bounds, give an upper bound on the probability that this new algorithm produces an incorrect result.

Solution:

Let X_i be the indicator of the i-th iteration.
$X_i = 1$ if the i-th iteration computes the correct answer;
$X_i = 0$ if the i-th iteration computes the wrong answer.
$X = \sum_{i=1}^{n} X_i$, $E[X_i] = 2/3$, so $E[X] = 2n/3$.
According to Chernoff bounds,
$P(X \leq n/2) = P(E[X] \leq (1 - 1/4)) \leq e^{-n/48} < (0.99)^n$.