UM–SJTU Joint Institute

VE477 Intro to Algorithms

Homework 5

Wang, Tianze
515370910202

Question 1 Partition Problem

(1) Definition

It is the task of deciding whether a given multiset S of positive integers can be partitioned into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of the numbers in S_2.

$$S = S_1 \cup S_2 \ \wedge \ S_1 \cap S_2 = \varnothing \ \wedge \ \sum S_1 = \sum S_2$$

(2) Simple Solution

No, it is not a good decision. For example, suppose we have a set like

$$\{1,\ 2,\ 97,\ 99\}$$

And we wish to partition it into two sets. Using this method will lead to a max set of 196, however if we partition it into

$$\{1, 2, 97\}, \{99\}$$

This will lead to a better solution, which is 100.

(3) Recursive Algorithm

Question 2 Critical Thinking

Here we use the idea as: The binary representative of a decimal number.

0-4 to 0-7

7 can be represented as 111 in binary. Thus 0 to 7 is 000_b to 111_b.

Input : 7
Output : a random integer between 0-7

```
1  Function generator(3):
2      i ← 3 ;
3      while i ≠ 0 do
4          j ← get an output from black box ;
           /* << means the operation of shift left                    */
5          if j = 0 or 1 then
6              b = 0 ;
7              a = (a << 1) + b ;
8              i − − ;
9          else if j = 2 or 3 then
10             b = 1 ;
11             a = (a << 1) + b ;
12             i − − ;
13         else
14             Continue;
15         end if
16     end while
17     return a
18 end
```

---

(1) 0-4 to common case

For the common case, we apply the same idea, but we need a judging condition to represent whether our result falls into the range of acceptance.

```
   Input  : n
   Output : a random integer between 0-n
1  Function generator(n):
2      A ← a binary number ;
3      while a>n do
4          i ← ⌈log_2 n⌉ ;
5          while i ≠ 0 do
6              j ← get an output from black box ;
               /* << means the operation of shift left                    */
7              if j = 0 or 1 then
8                  b = 0 ;
9                  a = (a << 1) + b ;
10                 i − − ;
11             else if j = 2 or 3 then
12                 b = 1 ;
13                 a = (a << 1) + b ;
14                 i − − ;
15             else
16                 Continue;
17             end if
18         end while
19     end while
20 end
```