COMP2201 – Discrete Mathematics
Basics Of Counting - Tutorial solutions

1. A small country (9 million people) is installing a new state-of-the-art telephone system. How many digits are necessary to allow sufficient phone numbers? Discuss the assumptions you have made, as well as the justification for your answer

Solution

Assuming a phone number can begin with any digit then need the smallest r such that

$10^r \geq 9$ million

$\log_{10} 9{,}000{,}000 = 6.9$

Therefore need at least 7 digits

2. Many people are paid every other week. Show that there will always be a month in which these people receive three paycheques.

Solution

52 weeks in a year

Biweekly cheques will be sent 26 times a year

There are 12 months in a year

Using Pigeon hole principle

If n objects are distributed into k boxes, then at least one box must contain $\left\lceil \frac{n}{k} \right\rceil$ objects.

Therefore,

$\left\lceil \frac{26}{12} \right\rceil = 3 \text{ cheques must be deposited in one month}$

3. **Use the inclusion-exclusion to calculate the number of bit strings of length 7 that do not contain a sequence of five consecutive 1s.**

Solution

Inclusion-Exclusion Principle

$$|A_1 \cup A_2 \cup A_3| = |A_1| + |A_2| + |A_3| - (|A_1 \cap A_2| + |A_1 \cap A_3| + |A_2 \cap A_3|) + |A_1 \cap A_2 \cap A_3|$$

Let A_1 = set of all bit strings of length 7 with 1's in the first five positions

Let A_2 = set of all bit strings of length 7 with 1's in positions 2-6

Let A_3 = set of all bit strings of length 7 with 1's in positions 3-7

Now

$$|A_1| = |A_2| = |A_3| = 2^2 = 4$$

$$|A_1 \cap A_2 \cap A_3| = 1$$

$$|A_1 \cap A_2| = |A_2 \cap A_3| = 2$$

$$|A_1 \cap A_3| = 1$$

Therefore, the number of bit strings of length 7 that contains a sequence of 5 consecutive 1's is given by

$$|A_1 \cup A_2 \cup A_3| = (4+4+4)-(2+2+1)+1=8$$

The total number of 7 bits strings is given by

$2^7 = 128$

Therefore the total number of bit strings of length 7 that does not contain a sequence of 5 consecutive 1's is given by

128 - 8 = 120

4. **Use the inclusion-exclusion to calculate the number of bit strings of length 9 that begin with two 0s, have eight consecutive 0s, or end with a 1 bit.**

Solution

Inclusion-Exclusion Principle

$$|A_1 \cup A_2 \cup A_3| = |A_1| + |A_2| + |A_3| - (|A_1 \cap A_2| + |A_1 \cap A_3| + |A_2 \cap A_3|) + |A_1 \cap A_2 \cap A_3|$$

Let A_1 = set of all bit strings of length 9 that begin with two 0s
Let A_2 = set of all bit strings of length 9 that have eight consecutive 0s
Let A_3 = set of all bit strings of length 9 that end with a 1 bit

Now

$|A_1| = 2^7 = 128$
$|A_2| = 2 + 2 - 1 = 3$
$|A_3| = 2^8 = 256$

$|A_1 \cap A_2 \cap A_3| = 1$

$|A_1 \cap A_2| = 2$

$|A_2 \cap A_3| = 1$

$|A_1 \cap A_3| = 2^6 = 64$

Therefore, the number of bit strings of length 9 that begin with two 0s, have eight consecutive 0s, or end with a 1 bit is given by

$$|A_1 \cup A_2 \cup A_3| = 128 + \mathbf{3} + 256 - (2 + 1 + 64) + 1 = 32\mathbf{1}$$

5. By constructing a Tree Diagram, determine how many 3-combinations are there of six objects (selecting all 6 objects)?

Ans. 6!/3!3! or ${}^6C_3{}^3C_3$ or 20

Solution

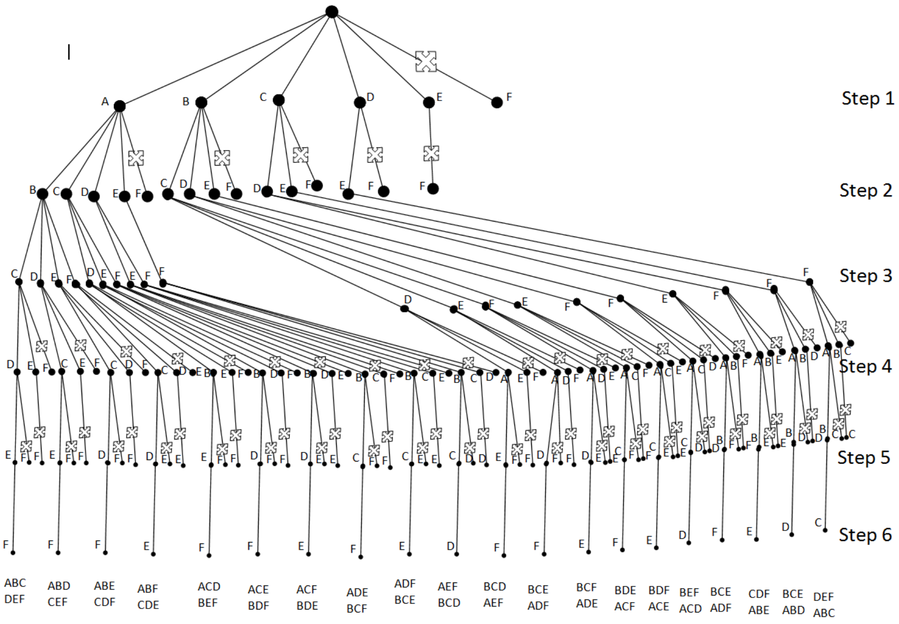
abc def
acd bef
bcd aef
def abc
bef acd
aef bcd

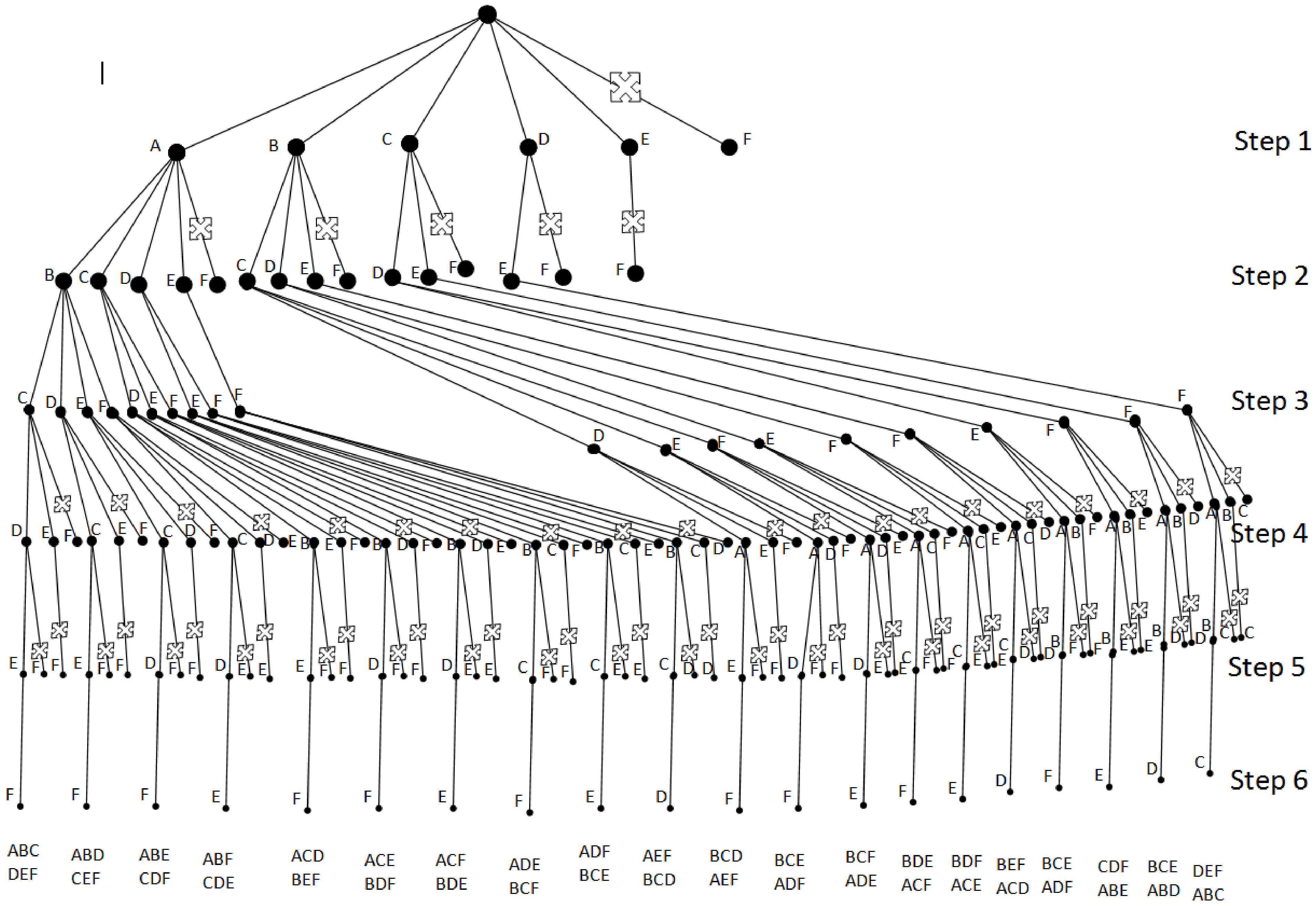
abd cef
ace bdf
bce adf
cef abd
bdf ace
adf bce

abe cdf
acf bdf
bcf ade
cdf abe
bdf acf
ade bcf

abf cde

cde abf




A
B
C
D
E
F
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
ABC
DEF
ABD
CEF
ABE
CDF
ABF
CDE
ACD
BEF
ACE
BDF
ACF
BDE
ADE
BCF
ADF
BCE
AEF
BCD
BCD
AEF
BCE
ADF
BCF
ADE
BDE
ACF
BDF
ACE
BEF
ACD
BCE
ADF
CDF
ABE
BCE
ABD
DEF
ABC

6. The following is a Python function *fib_seq* (from COMP1126) that prints Fibonacci series up to a number n.

```
def fib_seq(n):    # write Fibonacci series up to n
    a, b = 0, 1
    while a < n:
        print a,
        a, b = b, a+b
```

Write a Python function *fib_val* that receives positive integer argument *n* and returns the corresponding value in the Fibonacci sequence,

Solution

```
def fib_val(n):
   if n == 1 or n == 2:
        return 1
   else:
        return fib_val(n-1) + fib_val(n-2)
```

7. Write a Java method *lexic_less* that takes two strings *x* and *y* and returns if *x* is lexicographically less than *y*.

Solution

```
public boolean lexic_less (String x, String y) {
   int min = (x.length() <= y.length())  ?  x.length() :  y.length();
   int i;

   boolean result = true;
   for (i = 0; i < min; i++)
      if (x.charAt(i) > y.charAt(i)) {
         result = false;
         break;
      }
      else if (x.charAt(i) < y.charAt(i))
         break;
   if ((x.length() > y.length()) && (i >= min))
      result = false;
   return result;
}
```