Lab 01

I. El-Shaarawy, & M. Touny

October 22, 2023

Contents

Exercises

Design Thinking

- **Definition**. well-define special keywords like *exhaustive search*, *floor*, *square*.
- **Break**. Break the big problem into subproblems.
- **Partial Progress**. Solve subproblems or relaxed versions.
- **Concrete examples**. Try on concrete cases.
- **Generalize.** Spot the pattern generalizable on any case conforming to the general definition.
- **Connect Ideas**. Figure whether the different ideas and solutions can combined.

Algorithmic Hints

- Are there redundant computations?

1.1.4

Hints

- You are given a square number n. Given some integer k, How can we verify it is the root?
- Follow the exhaustive search strategy, to find the root of n.
- You are given a real number r. Given some integer k, How can we verify it is the floor of r?
- Follow the exhaustive search strategy, to find the floor of n.
- Combine all previous hints to find a unique definition of $\lfloor \sqrt{n} \rfloor$.
- Follow the exhaustive search strategy, to solve the main problem.

Solution

```
for i in n-1 .. 0
  if (i)^2 <= n
    return i
```

1.1.8

Hints

- Try this case on concrete examples like $m = 2$ and $n = 3$.
- Why $m \mod n = m$ when $m < n$?

- Recall the definition of $\mod$. What are the possible ranges of $x \mod n$ for any integer x?

Solution

It shall swap them as $r = m \mod n = m$ when $m < n$.

Only once. Given $m > n$, Necessarily $n > m \mod n$.

1.2.5

Hints

- Convert a concrete decimal number to binary. Observe how the right most digit from the binary representation is obtained.
- Given a binary representation, What is the number we divide on it, so that the quotient eliminate the right most digit?
- Follow the *Decrease and Conquer* strategy, with the above two hints, to solve the problem.

Solution

```
DecToBin(n):
  # input: integer n
  # output: binary representation as a list

  # binary representation
  l = [ ]

  while n != 0:
    # kth digit from right to left
    b.appendLeft( n % 2 )

    # remove the rightmost digit
    # division output is an integer
    n = n/2
```

Algorithm 1 Convert to the binary representation of a given integer

Require: input is integer n
Ensure: output is a list of binary digits

1: $l \leftarrow [\,]$
2: **while** $n \neq 0$ **do**
3: $\quad l \leftarrow [n \mod 2] \cup l$.
4: $\quad n \leftarrow n/2$
5: **end while**

1.2.9

Hint

- Are there duplicated computations?
- Are there pairs tested twice?
- Observe $|a-b| = |b-a|$.
- If we checked all elements with $A[i]$, Do we need to test $A[j]$ with $A[i]$?

Solution

```
MinDistance( A ):
  # input: array of size n
  # output: minimum distance between two distinct elements

  dmin = infinity
  for i in 0 .. n-1:
    for j in i+1 .. n-1:
      dis = | A[i] - A[j] |
      if dis < dmin:
        dmin = dis
```

Algorithm 2 Find the minimum distance between two distinct elements in an array

Require: input is array $A[0..n-1]$ of numbers
Ensure: output is the minimum distance between any two distinct elements

$dmin \leftarrow \infty$
for $i = 0$ to $n-1$ **do**
  for $j = i+1$ to $n-1$ **do**
    $dis \leftarrow |A[i] - A[j]|$
    if $dis < dmin$ **then**
      $dmin \leftarrow dis$
    end if
  end for
end for
return dmin

1.3.1

Hints

- if $A[i] == A[j]$ which index shall be counted? What can we conclude about S?

Solution

a. Tedious to typeset.

b. No. Observe counting only happens when strictly $i < j$. If $A[i] == A[j]$ then the code counts $A[i]$ not $A[j]$. Therefore $A[i]$ shall succeed $A[j]$. In fact equal cells are reversed in the sorted array.

c. No. It does not modify array A but output is a different array S.

1.3.10

FAILED TO SOLVE.

1.4.2

Hint

- For ascendingly ordered array A, Is it possible for the target value t to exist in $A[i..n-1]$ given the fact $t > A[i]$?
- Use the above hint to prune the search space.
- Which index of the array you think shall prune the greatest search space.

Solution

For target value t:

a. Access some element x in the array. If $t \neq x$, We can ignore searching in the right/left side of x.

b. While linear scanning, Terminate the algorithm earlier once some $A[i] > t$.

1.4.10

Hints

- Is it possible for two strings to be anagrams in case they different lengths?
- Is it possible for two strings to be anagrams if one of them has a character not present in the other?
- You can convert a character to its corresponding ascii number. Use that for a cheaper data strucutre.
- the ascii number corresponds to an index.

Solution

Two strings are *anagrams* if and only if they have the same count of characters.

```
AreStringsAnagrams(A, B):
  # input two strings
```

```
# output True if anagrams and False otherwise

# if lengths are not the same, then not anagrams
if length(A) != length(B):
  return False

# initialize characters counts to zeros for both strings
A_chCount = B_chCount = [ 0 ] * 26

# Count characters in both strings
for ch in A:
  A_chCount[ int(ch) ] = A_chCount[ int(ch) ] + 1

for ch in B:
  B_chCount[ int(ch) ] = B_chCount[ int(ch) ] + 1

# Anagrams if and only if characters count is exactly the same
return A_chCount == B_chCount
```

Algorithm 3 Detect whether two strings are anagrams

Require: input is two strings
Ensure: output True if anagrams and False otherwise

if $|A| \neq |B|$ **then**
  return **false**
end if

$A_{count} \leftarrow B_{count} \leftarrow [\,] \times 26$

for $ch \in A$ **do**
  $A_{count}[ch] \leftarrow A_{count}[ch] + 1$
end for

for $ch \in B$ **do**
  $B_{count}[ch] \leftarrow B_{count}[ch] + 1$
end for

return $A_{count} == B_{count}$