LIVE 6: Dynamic Programming + inbuilt-DS

- Focus: Dynamic programming example , where to use which DS in Python
- Prereq: Previous code-sessions + Basic knowledge of inbuilt DS in Python(list, tuple, dict, set)
- Reference for basics:
  - https://docs.python.org/3/tutorial/datastructures.html# (https://docs.python.org/3/tutorial/datastructures.html#)
  - https://www.geeksforgeeks.org/inbuilt-data-structures-python/ (https://www.geeksforgeeks.org/inbuilt-data-structures-python/)

Problem 4: Regex matching problem

- "?" matches a single character
- "*" matches zero or more charcters
- Given a pattern(p) and a string(s), does p match s?
- examples:
  - TRUE: ("*", "ab"), ("?a" ,"ba"), ("?a" ,"aa"), ("a*" ,"a")
  - FALSE: ("*a", "ab"), ("?a" ,"baa"), ("?a" ,"a"), ("a*" ,"ba")
- Very popular interview question at product-based companies for SDEs.
- Small variations of this are often used in interviews
- Any suggestions?

In [238]:

```
# Handle all cases of recursion thoroughly.

def isMatch(p,s):

    print(p,s) # print statemnt for debugging

    # boundary cases of recursion
    if p == s:
        return True

    if p == "*":
        return True

    if p == "" or s == "":
        return False

    # recursion case-1
    if p[0] == s[0] or p[0] == '?':
        return isMatch(p[1:], s[1:])

    # recursion-case-2
    if p[0] == '*':
        return ( isMatch( p[1:], s) or isMatch( p, s[1:]))

    # last case: if p[0] is a character
    if  p[0] != s[0]:
        return False;
```

In [239]:

```
print(isMatch("*","ab"))
```

```
* ab
True
```

In [240]:

```
print(isMatch("?a" ,"baa"))
```

```
?a baa
a aa
 a
False
```

In [241]:

```
print(isMatch("a*" ,"ba"))
```

```
a* ba
False
```

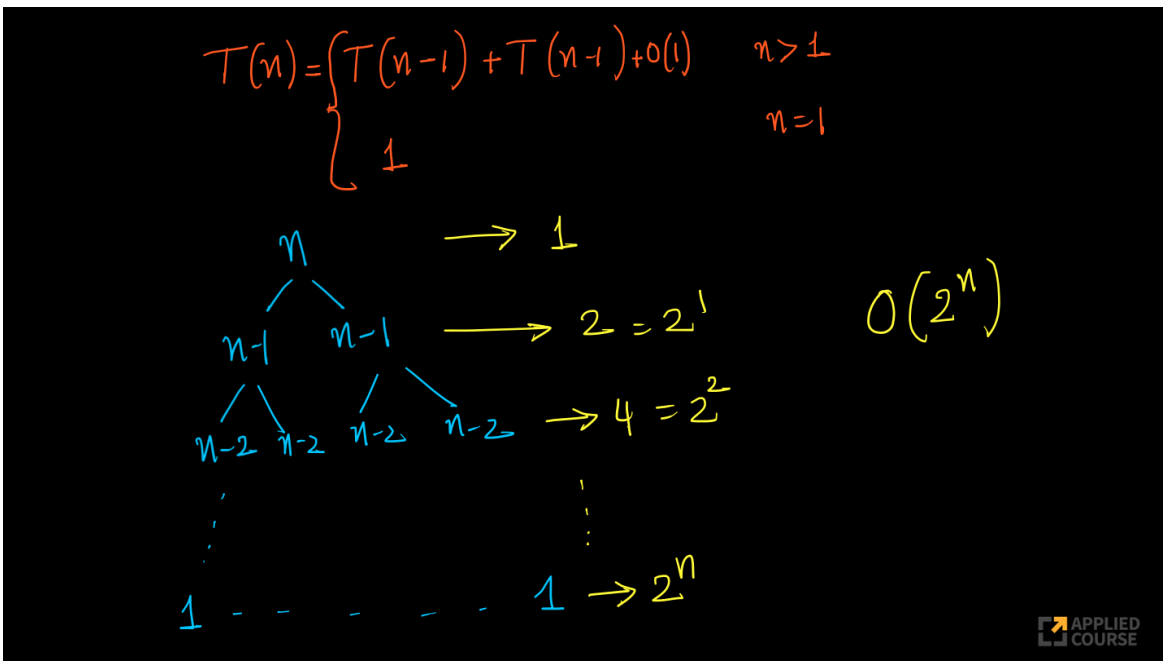
In [242]:

```
# Worst case Time Complexity: T(n) = T(n-1) + T(n-1) = 2* T(n-1)

# => T(n) = 0(2^n) [as shown below]


from IPython.display import Image
Image(url= "https://i.imgur.com/dse47H3.png")
```

Out[242]:

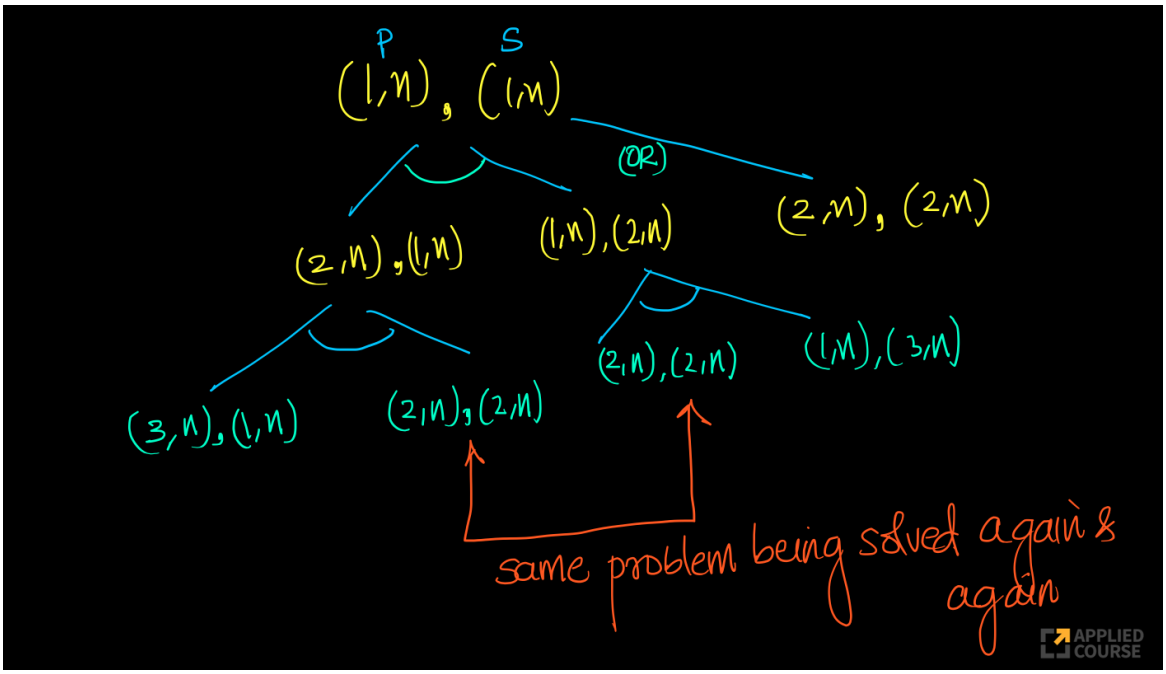


In [243]:

```
# Can we do better?

Image(url= "https://i.imgur.com/Rx6tN8a.png")
```

Out[243]:



In [244]:

```
# Overlapping sub-problems:
# Why not store solutions to already solved problems in a dict (dp)
# Dynamic programming: Recursion + overlapping subproblems


# We discussed this in our course when we learn back-propogation in Deep-Learnin
g.
```

In [269]:

```
# top-down DP using recursion and memoization (dp)



def isMatchDP(pattern, string, pattern_start, string_start, dp ):

    #print(pattern_start, string_start);

    if  (pattern_start, string_start) in dp:
        return dp[(pattern_start, string_start)]

    p = pattern[pattern_start:];
    s = string[string_start:];


    print(p,s) # print statemnt for debugging

    # boundary cases of recursion
    if p == s:
        dp[(pattern_start, string_start)] = True;
        return True

    if p == "*":
        dp[(pattern_start, string_start)] = True;
        return True

    if p == "" or s == "":
        dp[(pattern_start, string_start)] = False;
        return False

    # recursion case-1
    if p[0] == s[0] or p[0] == '?':
        tmp = isMatchDP(pattern, string, pattern_start+1, string_start+1, dp)
        d[(pattern_start, string_start)] = tmp;
        return tmp;

    # recursion-case-2
    if p[0] == '*':
        tmp = ( isMatchDP( pattern, string, pattern_start+1, string_start, dp) o
r isMatchDP( pattern, string, pattern_start, string_start+1, dp))
        dp[(pattern_start, string_start)] = tmp;
        return tmp;

    # last case: if p[0] is a character
    if  p[0] != s[0]:
        dp[(pattern_start, string_start)] = False;
        return False;
```

In [270]:

```
print(isMatchDP("*","ab",0,0, {}))
```

```
* ab
True
```

In [272]:

```
print(isMatchDP("?a" ,"baa",0,0,{}))
```

```
?a baa
a aa
 a
False
```

In [273]:

```
print(isMatchDP("?a*" ,"baa",0,0,{}))
```

```
?a* baa
a* aa
* a
True
```

2 Dimensional Dynamic programming vs 1-Dimensional DP

Additional Resources:

- https://www.geeksforgeeks.org/solve-dynamic-programming-problem/ (https://www.geeksforgeeks.org/solve-dynamic-programming-problem/)
- 50 practice problems: https://blog.usejournal.com/top-50-dynamic-programming-practice-problems-4208fed71aa3 (https://blog.usejournal.com/top-50-dynamic-programming-practice-problems-4208fed71aa3)

Python's inbuilt data-structures: List, Dict, Set, Tuple

- Assumption: basic understadning about these DS.
- Big practical question: Where to use which DS?

Key properties (recap):

- LIST: [1,2,3,4,5], ordered, indexable, iterable
- DICT: {'jack': 4098, 'sape': 4139}, O(1) search, <k,v> pairs, iterable, ordered(3.7+) [https://docs.python.org/3/whatsnew/3.7.html (https://docs.python.org/3/whatsnew/3.7.html)]
- TUPLE: (1,2,3), immutable, ordered-k-items,
- SET: {1,2,3}, unordered, iterable, no-duplicates.

" A DS is decided based on what operations you want to perform on it and how fast these operations are"

Lets learn: Which DS will you use in a given situation and why?

Q. Represent a 3-D point as a parameter to a function: f(3D-point)

Notice the shape parameter in NumPy

class numpy.ndarray(shape, dtype=float, buffer=None, offset=0, strides=None, order=None)[source] An array object represents a multidimensional, homogeneous array of fixed-size items. An associated data-type object describes the format of each element in the array (its byte-order, how many bytes it occupies in memory, whether it is an integer, a floating point number, or something else, etc.)

Arrays should be constructed using array, zeros or empty (refer to the See Also section below). The parameters given here refer to a low-level method (ndarray(…)) for instantiating an array.

For more information, refer to the numpy module and examine the methods and attributes of an array.

Parameters: (for the **new** method; see Notes below) shape : tuple of ints Shape of created array.

dtype : data-type, optional Any object that can be interpreted as a numpy data type.

buffer : object exposing buffer interface, optional Used to fill the array with data.

offset : int, optional Offset of array data in buffer

Q. Represent a 3-D point as a parameter to a function: f(3D-point) which changes the 3-D point's co-ordinates

In [179]:

```
def f(t):
    t[0] = t[0]+1;
    t[1] = t[1]+1;
    t[2] = t[2]+1;
    return;

t=(0,1,2);
print(t);

print(t[0])

f(t);

print(t)

# tuples are immutable
```

```
(0, 1, 2)
0

---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-179-be22e0c9180e> in <module>
     10 print(t[0])
     11
---> 12 f(t);
     13
     14 print(t)

<ipython-input-179-be22e0c9180e> in f(t)
      1 def f(t):
----> 2     t[0] = t[0]+1;
      3     t[1] = t[1]+1;
      4     t[2] = t[2]+1;
      5     return;

TypeError: 'tuple' object does not support item assignment
```

In [180]:

```
# lists are mutable and ordered.

def f(t):
    t[0] = t[0]+1;
    t[1] = t[1]+1;
    t[2] = t[2]+1;
    return;

t=[0,1,2];
print(t);

print(t[0])

f(t);

print(t)
```

```
[0, 1, 2]
0
[1, 2, 3]
```

Example: Mutable and immutable & Parameter passing

https://stackoverflow.com/a/986145 (https://stackoverflow.com/a/986145)

Q. If you were to design NumPy from scratch, which inbuilt DS would you use to design ndarray?

Q. How do you think data is internally stored in Numpy Arrays?

"Why are NumPy arrays efficient?

A NumPy array is basically described by metadata (notably the number of dimensions, the shape, and the data type) and the actual data. The data is stored in a homogeneous and contiguous block of memory, at a particular address in system memory (Random Access Memory, or RAM). This block of memory is called the data buffer. This is the main difference between an array and a pure Python structure, such as a list, where the items are scattered across the system memory. This aspect is the critical feature that makes NumPy arrays so efficient."

Source: https://ipython-books.github.io/45-understanding-the-internals-of-numpy-to-avoid-unnecessary-array-copying/ (https://ipython-books.github.io/45-understanding-the-internals-of-numpy-to-avoid-unnecessary-array-copying/)

Q. You want to write the program to obtain various combinations. How do you take the input?

Refer: https://docs.python.org/2/library/itertools.html#itertools.combinations (https://docs.python.org/2/library/itertools.html#itertools.combinations)

itertools.combinations(iterable, r)

Return r length subsequences of elements from the input iterable. Combinations are emitted in lexicographic sort order. So, if the input iterable is sorted, the combination tuples will be produced in sorted order. Elements are treated as unique based on their position, not on their value. So if the input elements are unique, there will be no repeat values in each combination.

In [6]:

```
import itertools

res = itertools.combinations([1,2,3,4,5],2)

print(type(res))
```

```
<class 'itertools.combinations'>
```

Q. How would you build a TinyURL service? [https://tinyurl.com/ (https://tinyurl.com/)]

- Popular interview question

Dict/Hashtable has <K,V> pairs

Key = short-URL Value = Long-URL

Key: random variable length alphabetic suffix [tinyurl.com/tgwxyc]

In [188]:

```
import random
import string

d = {};

# given a long URL, get a short URL
def getShortURL(longURL):
    # length = random value in 6-10
    l = random.randint(6,10);
    print(l);

    # generate random characters into a string of length l
    chars = string.ascii_lowercase
    shortURL = ''.join(random.choice(chars) for i in range(l))
    print(shortURL);

    # check if this string is already present in dict d
    if shortURL in d:
        return getShortURL(longURL);
    else:
        d[shortURL] = longURL;

    print(d)
    r = "https://www.shortURL.com/"+shortURL
    return r;
```

In [189]:

```
print(getShortURL("www.appliedaicourse.com"))
```

```
8
iftgtrsu
{'iftgtrsu': 'www.appliedaicourse.com'}
https://www.shortURL.com/iftgtrsu
```

In [190]:

```
print(getShortURL("www.appliedaicourse.com"))
```

```
7
mxzmuis
{'iftgtrsu': 'www.appliedaicourse.com', 'mxzmuis': 'www.appliedaicou
rse.com'}
https://www.shortURL.com/mxzmuis
```

In [191]:

```
def getLongURL(shortURL):

    # extarct key from URL https://www.shortURL.com/mxzmuis ---> mxzmuis
    k = shortURL[25:];
    print(k)

    return d[k];
```

In [192]:

```
print(getLongURL("https://www.shortURL.com/mxzmuis"))
```

```
mxzmuis
www.appliedaicourse.com
```

In [193]:

```
print(getLongURL("https://www.shortURL.com/abcdef"))
```

```
abcdef

-------------------------------------------------------------------
-------
KeyError                                  Traceback (most recent cal
l last)
<ipython-input-193-a1641eeac07b> in <module>
----> 1 print(getLongURL("https://www.shortURL.com/abcdef"))

<ipython-input-191-049da4c13feb> in getLongURL(shortURL)
      5     print(k)
      6
----> 7     return d[k];

KeyError: 'abcdef'
```

In [196]:

```
# Handle Errors and return None

def getLongURL(shortURL):

    # extarct key from URL https://www.shortURL.com/mxzmuis ---> mxzmuis
    k = shortURL[25:];
    print(k)

    if k in d:
        return d[k];
    else:
        return None;
```

In [197]:

```
print(getLongURL("https://www.shortURL.com/abcdef"))
```

```
abcdef
None
```

In [198]:

```
print(getLongURL("https://www.shortURL.com/mxzmuis"))
```

```
mxzmuis
www.appliedaicourse.com
```

IDEA: We have dict 'd' and these two functions(getShortURL, getLongURL) that opearete on 'd'. Why not put all these three in a single logical block?

- That's where "Classes" come in! Next session: Basics of OOP in Python (for ML)
- We will build a ShortURL class.

Q. Find common elements in 3 lists

l1 = [1,2,3,4]
l2 = [3,4,5]
l3 = [1,4,6,7,8]

result = [4]

In [201]:

```
l1 = [1,2,3,4]
l2 = [3,4,5]
l3 = [1,4,6,7,8]

s1 = set(l1);
s2 = set(l2);
s3 =set(l3);

result = list((s1.intersection(s2)).intersection(s3))

print(result)

# Simple application of Sets
```

```
[4]
```

Exercise: Find all documents which contain search keywords and phrases.

In [ ]:

```

```