Ch25-DynamicProgramming

November 2, 2021

1 Dynamic Programming (DP)

- https://www.cs.cmu.edu/~avrim/451f09/lectures/lect1001.pdf
- https://www.geeksforgeeks.org/overlapping-subproblems-property-in-dynamic-programming-dp-1/
- commonly used and powerful optimization technique that allows you to solve many different types of problemns in time $O(n^2)$ or $O(n^3)$ for which a naive approach would take exponential time
- two main properties of a problem that warrents DP solution:
  1. Overlapping Subproblems
  2. Optimal Substructures

1.1 Overlapping Subproblems

- problem combines solutions from many overlapping sub-problems
- DP is not useful when there are no common (overlapping) subproblems
- computed solutions to sub-problems are stored in a look-up table to avoid recomputation
- slighlty different from **Divide and Conquer** technqiue
  - divide the problems into smaller non-overlapping subproblems and solve them independently
  - e.g.: merge sort and quick sort

1.2 Optimal Substructures

- optimal solution of the given problem can be obtained by using optimal solutions of its subproblems

1.3 2 Types of DP solutions

1.4 1. Top-Down (Memoization)

- based on the Latin word memorandum, meaning "to be remembered"
- similar to word memorization, its a technique used in coding to improve program runtime by memorizing intermediate solutions
- using dict type lookup data structure, one can memorize intermediate results of subproblems
- tpically recursion use top-down approach

1.4.1 Process

- start solving the given problem by breaking it down

- first check to see if the given problem has been solved already
  - if so, return the saved answer
  - if not, solve it and save the answer

1.5 2. Bottom-Up (Tabulation)

- start solving from the trivial subproblem
- store the results in a table/list/array
- move up towards the given problem by using the results of subproblems
- typically iterative solutions uses bottom-up approach

1.5.1 simple recursive fib function

- recall, fibonacci definition is recursive and has many common/overlapping subproblems

[1]:
```
count = 0
def fib(n):
    global count
    count += 1
    if n <= 2:
        return 1
    f = fib(n-1) + fib(n-2)
    return f

n=30 #40, 50? takes a while
print("fib at {}th position = {}".format(n, fib(n)))
print("fib function count = {}".format(count))
```

```
fib at 30th position = 832040
fib function count = 1664079
```

[2]:
```
# let's add looging to the function to vidualize the function call on stack

import inspect

def stack_depth():
    return len(inspect.getouterframes(inspect.currentframe())) - 1
```

[4]:
```
def fib_stack(n):
    print(f'{" "*stack_depth()}fib_stack({n}) called')
    if n <= 2:
        return 1
    f = fib_stack(n-1) + fib_stack(n-2)
    return f

n=10 # big number takes longer and output will not look as great
print("fib at {}th position = {}".format(n, fib_stack(n)))
```

```
fib_stack(10) called
 fib_stack(9) called
  fib_stack(8) called
   fib_stack(7) called
    fib_stack(6) called
     fib_stack(5) called
      fib_stack(4) called
       fib_stack(3) called
        fib_stack(2) called
        fib_stack(1) called
       fib_stack(2) called
      fib_stack(3) called
       fib_stack(2) called
       fib_stack(1) called
     fib_stack(4) called
      fib_stack(3) called
       fib_stack(2) called
       fib_stack(1) called
      fib_stack(2) called
    fib_stack(5) called
     fib_stack(4) called
      fib_stack(3) called
       fib_stack(2) called
       fib_stack(1) called
      fib_stack(2) called
     fib_stack(3) called
      fib_stack(2) called
      fib_stack(1) called
   fib_stack(6) called
    fib_stack(5) called
     fib_stack(4) called
      fib_stack(3) called
       fib_stack(2) called
       fib_stack(1) called
      fib_stack(2) called
     fib_stack(3) called
      fib_stack(2) called
      fib_stack(1) called
    fib_stack(4) called
     fib_stack(3) called
      fib_stack(2) called
      fib_stack(1) called
     fib_stack(2) called
  fib_stack(7) called
   fib_stack(6) called
    fib_stack(5) called
     fib_stack(4) called
      fib_stack(3) called
```

```
      fib_stack(2) called
      fib_stack(1) called
     fib_stack(2) called
    fib_stack(3) called
     fib_stack(2) called
     fib_stack(1) called
   fib_stack(4) called
    fib_stack(3) called
     fib_stack(2) called
     fib_stack(1) called
    fib_stack(2) called
  fib_stack(5) called
   fib_stack(4) called
    fib_stack(3) called
     fib_stack(2) called
     fib_stack(1) called
    fib_stack(2) called
   fib_stack(3) called
    fib_stack(2) called
    fib_stack(1) called
fib_stack(8) called
 fib_stack(7) called
  fib_stack(6) called
   fib_stack(5) called
    fib_stack(4) called
     fib_stack(3) called
      fib_stack(2) called
      fib_stack(1) called
     fib_stack(2) called
    fib_stack(3) called
     fib_stack(2) called
     fib_stack(1) called
   fib_stack(4) called
    fib_stack(3) called
     fib_stack(2) called
     fib_stack(1) called
    fib_stack(2) called
  fib_stack(5) called
   fib_stack(4) called
    fib_stack(3) called
     fib_stack(2) called
     fib_stack(1) called
    fib_stack(2) called
   fib_stack(3) called
    fib_stack(2) called
    fib_stack(1) called
 fib_stack(6) called
  fib_stack(5) called
```

```
                                fib_stack(4) called
                                 fib_stack(3) called
                                  fib_stack(2) called
                                  fib_stack(1) called
                                 fib_stack(2) called
                                fib_stack(3) called
                                 fib_stack(2) called
                                 fib_stack(1) called
                               fib_stack(4) called
                                fib_stack(3) called
                                 fib_stack(2) called
                                 fib_stack(1) called
                                fib_stack(2) called
fib at 10th position = 55
```

1.5.2 theoritical computational complexity

- Time Complexity: $T(n)$ = time to calculate Fib(n-1) + Fib(n-2) + time to add them: O(1)
- using Big-Oh (O) notation for upper-bound:
  - $T(n) = T(n-1) + T(n-2) + O(1)$
  - $T(n) = O(2^{n-1}) + O(2^{n-2}) + O(1)$
  - $T(n) = O(2^n)$

  precisely
  - $T(n) = O(1.6)^n$
    - 1.6… is called golden ratio - https://www.mathsisfun.com/numbers/golden-ratio.html
- Space Complexity = $O(n)$ due to call stack

[5]:
```python
#print(globals())
import timeit
print(timeit.timeit('fib(30)', number=1, globals=globals()))
# big difference between 30 and 40
```

```
0.2583559870000016
```

1.5.3 memoized recursive fib function

[6]:
```python
count = 0
def MemoizedFib(memo, n):
    global count
    count += 1
    if n <= 1:
        return 1
    if n in memo:
        return memo[n]
    memo[n] = MemoizedFib(memo, n-1) + MemoizedFib(memo, n-2)
    return memo[n]
```

[7]:
```python
memo = {}
n=1000 #try 40, 50, 60, 100, 500, 10000, ...
print("fib at {}th position = {}".format(n, MemoizedFib(memo, n)))
print("fib function called {} times.".format(count))
```

```
fib at 1000th position = 70330367711422815821835254877183549770181269836358732742604905087154537118196933579742249494562611733487750449241765991088186363265450223647106012053374121273867339111198139373125598767690091902245245323403501
fib function called 1999 times.
```

[8]:
```python
# let's time the MemoizedFib(1000)
import timeit
memo = {}
n=1000
print(timeit.timeit('MemoizedFib(memo, n)', number=1, globals=globals()))
```

```
0.0007638900000017657
```

1.6 using function decorator @cache

- no need to write our own caching mechanism
- @cache is new in Python 3.9; so update python to 3.9 if necessary
- NOTE - works only for subsequent calls not the first time recrusion is called!

[9]:
```
! which python
```

```
/Users/rbasnet/miniconda3/envs/py/bin/python
```

[10]:
```
! python --version
```

```
Python 3.9.7
```

[11]:
```
%%bash
# update silently without prompting
conda update python -y
```

```
Collecting package metadata (current_repodata.json): …working… done
Solving environment: …working… done

## Package Plan ##

  environment location: /Users/rbasnet/miniconda3/envs/py

  added / updated specs:
    - python


The following packages will be downloaded:
```

```
    package                    |            build
    ---------------------------|-----------------
    anyio-2.2.0                |   py39hecd8cb5_1         125 KB
    brotlipy-0.7.0             |py39h9ed2024_1003         333 KB
    cryptography-35.0.0        |   py39h2fd3fbb_0         1.1 MB
    jupyter_server-1.4.1       |   py39hecd8cb5_0         312 KB
    notebook-6.4.5             |   py39hecd8cb5_0         4.2 MB
    pysocks-1.7.1              |   py39hecd8cb5_0          31 KB
    sniffio-1.2.0              |   py39hecd8cb5_1          15 KB
    ------------------------------------------------------------
                                           Total:         6.1 MB

The following NEW packages will be INSTALLED:

  importlib_metadata pkgs/main/noarch::importlib_metadata-4.8.1-hd3eb1b0_0

The following packages will be REMOVED:

  backports-1.0-py_2
  backports.functools_lru_cache-1.6.4-pyhd8ed1ab_0
  chardet-4.0.0-py39h6e9494a_1
  pandoc-2.15-h0d85af4_0
  python_abi-3.9-2_cp39
  requests-unixsocket-0.2.0-py_0
  websocket-client-0.57.0-py39h6e9494a_4

The following packages will be UPDATED:

  appnope            conda-forge::appnope-0.1.2-py39h6e949~ -->
pkgs/main::appnope-0.1.2-py39hecd8cb5_1001
  brotlipy           conda-forge::brotlipy-0.7.0-py39h89e8~ -->
pkgs/main::brotlipy-0.7.0-py39h9ed2024_1003
  ca-certificates    conda-forge::ca-certificates-2021.10.~ --> pkgs/main::ca-
certificates-2021.10.26-hecd8cb5_2
  charset-normalizer conda-forge::charset-normalizer-2.0.0~ -->
pkgs/main::charset-normalizer-2.0.4-pyhd3eb1b0_0
  idna                   conda-forge::idna-3.1-pyhd3deb0d_0 -->
pkgs/main::idna-3.2-pyhd3eb1b0_0
  json5                 conda-forge::json5-0.9.5-pyh9f0ad1d_0 -->
pkgs/main::json5-0.9.6-pyhd3eb1b0_0
  pexpect            conda-forge::pexpect-4.8.0-pyh9f0ad1d~ -->
pkgs/main::pexpect-4.8.0-pyhd3eb1b0_3
  ptyprocess         conda-forge::ptyprocess-0.7.0-pyhd3de~ -->
pkgs/main::ptyprocess-0.7.0-pyhd3eb1b0_2
  pyopenssl          conda-forge::pyopenssl-21.0.0-pyhd8ed~ -->
pkgs/main::pyopenssl-21.0.0-pyhd3eb1b0_1
  pyrsistent         conda-forge::pyrsistent-0.17.3-py39h8~ -->
pkgs/main::pyrsistent-0.18.0-py39h9ed2024_0
```

```
  send2trash         conda-forge::send2trash-1.8.0-pyhd8ed~ -->
pkgs/main::send2trash-1.8.0-pyhd3eb1b0_1

The following packages will be SUPERSEDED by a higher-priority channel:

  anyio              conda-forge::anyio-3.3.4-py39h6e9494a~ -->
pkgs/main::anyio-2.2.0-py39hecd8cb5_1
  argon2-cffi        conda-forge::argon2-cffi-21.1.0-py39h~ -->
pkgs/main::argon2-cffi-20.1.0-py39h9ed2024_1
  async_generator    conda-forge::async_generator-1.10-py_0 -->
pkgs/main::async_generator-1.10-pyhd3eb1b0_0
  attrs              conda-forge::attrs-21.2.0-pyhd8ed1ab_0 -->
pkgs/main::attrs-21.2.0-pyhd3eb1b0_0
  babel               conda-forge::babel-2.9.1-pyh44b312d_0 -->
pkgs/main::babel-2.9.1-pyhd3eb1b0_0
  backcall           conda-forge::backcall-0.2.0-pyh9f0ad1~ -->
pkgs/main::backcall-0.2.0-pyhd3eb1b0_0
  bleach             conda-forge::bleach-4.1.0-pyhd8ed1ab_0 -->
pkgs/main::bleach-4.0.0-pyhd3eb1b0_0
  certifi            conda-forge::certifi-2021.10.8-py39h6~ -->
pkgs/main::certifi-2021.10.8-py39hecd8cb5_0
  cffi               conda-forge::cffi-1.14.6-py39hb71fe58~ -->
pkgs/main::cffi-1.14.6-py39h2125817_0
  cryptography       conda-forge::cryptography-35.0.0-py39~ -->
pkgs/main::cryptography-35.0.0-py39h2fd3fbb_0
  debugpy            conda-forge::debugpy-1.4.1-py39h9fcab~ -->
pkgs/main::debugpy-1.4.1-py39h23ab428_0
  decorator          conda-forge::decorator-5.1.0-pyhd8ed1~ -->
pkgs/main::decorator-5.1.0-pyhd3eb1b0_0
  defusedxml         conda-forge::defusedxml-0.7.1-pyhd8ed~ -->
pkgs/main::defusedxml-0.7.1-pyhd3eb1b0_0
  entrypoints        conda-forge/noarch::entrypoints-0.3-p~ -->
pkgs/main/osx-64::entrypoints-0.3-py39hecd8cb5_0
  importlib-metadata conda-forge::importlib-metadata-4.8.1~ -->
pkgs/main::importlib-metadata-4.8.1-py39hecd8cb5_0
  ipykernel          conda-forge::ipykernel-6.4.2-py39h71a~ -->
pkgs/main::ipykernel-6.4.1-py39hecd8cb5_1
  ipython            conda-forge::ipython-7.28.0-py39h71a6~ -->
pkgs/main::ipython-7.27.0-py39h01d92e1_0
  ipython_genutils   conda-forge::ipython_genutils-0.2.0-p~ -->
pkgs/main::ipython_genutils-0.2.0-pyhd3eb1b0_1
  jedi               conda-forge::jedi-0.18.0-py39h6e9494a~ -->
pkgs/main::jedi-0.18.0-py39hecd8cb5_1
  jinja2             conda-forge::jinja2-3.0.2-pyhd8ed1ab_0 -->
pkgs/main::jinja2-3.0.1-pyhd3eb1b0_0
  jsonschema         conda-forge::jsonschema-4.1.2-pyhd8ed~ -->
pkgs/main::jsonschema-3.2.0-pyhd3eb1b0_2
  jupyter_client     conda-forge::jupyter_client-7.0.6-pyh~ -->
```

```
pkgs/main::jupyter_client-7.0.1-pyhd3eb1b0_0
  jupyter_core       conda-forge::jupyter_core-4.9.1-py39h~ -->
pkgs/main::jupyter_core-4.8.1-py39hecd8cb5_0
  jupyter_server     conda-forge/noarch::jupyter_server-1.~ -->
pkgs/main/osx-64::jupyter_server-1.4.1-py39hecd8cb5_0
  jupyterlab         conda-forge::jupyterlab-3.2.1-pyhd8ed~ -->
pkgs/main::jupyterlab-3.1.7-pyhd3eb1b0_0
  jupyterlab_pygmen~ conda-forge::jupyterlab_pygments-0.1.~ -->
pkgs/main::jupyterlab_pygments-0.1.2-py_0
  jupyterlab_server  conda-forge::jupyterlab_server-2.8.2-~ -->
pkgs/main::jupyterlab_server-2.8.2-pyhd3eb1b0_0
  libsodium          conda-forge::libsodium-1.0.18-hbcb390~ -->
pkgs/main::libsodium-1.0.18-h1de35cc_0
  markupsafe         conda-forge::markupsafe-2.0.1-py39h89~ -->
pkgs/main::markupsafe-2.0.1-py39h9ed2024_0
  matplotlib-inline  conda-forge::matplotlib-inline-0.1.3-~ -->
pkgs/main::matplotlib-inline-0.1.2-pyhd3eb1b0_2
  mistune            conda-forge::mistune-0.8.4-py39h89e85~ -->
pkgs/main::mistune-0.8.4-py39h9ed2024_1000
  nbclassic          conda-forge::nbclassic-0.3.4-pyhd8ed1~ -->
pkgs/main::nbclassic-0.2.6-pyhd3eb1b0_0
  nbclient           conda-forge::nbclient-0.5.4-pyhd8ed1a~ -->
pkgs/main::nbclient-0.5.3-pyhd3eb1b0_0
  nbconvert          conda-forge::nbconvert-6.2.0-py39h6e9~ -->
pkgs/main::nbconvert-6.1.0-py39hecd8cb5_0
  nbformat           conda-forge::nbformat-5.1.3-pyhd8ed1a~ -->
pkgs/main::nbformat-5.1.3-pyhd3eb1b0_0
  nest-asyncio       conda-forge::nest-asyncio-1.5.1-pyhd8~ --> pkgs/main::nest-
asyncio-1.5.1-pyhd3eb1b0_0
  notebook           conda-forge/noarch::notebook-6.4.5-py~ -->
pkgs/main/osx-64::notebook-6.4.5-py39hecd8cb5_0
  openssl            conda-forge::openssl-1.1.1l-h0d85af4_0 -->
pkgs/main::openssl-1.1.1l-h9ed2024_0
  packaging          conda-forge::packaging-21.0-pyhd8ed1a~ -->
pkgs/main::packaging-21.0-pyhd3eb1b0_0
  pandocfilters      conda-forge/noarch::pandocfilters-1.5~ -->
pkgs/main/osx-64::pandocfilters-1.4.3-py39hecd8cb5_1
  parso               conda-forge::parso-0.8.2-pyhd8ed1ab_0 -->
pkgs/main::parso-0.8.2-pyhd3eb1b0_0
  pickleshare        conda-forge::pickleshare-0.7.5-py_1003 -->
pkgs/main::pickleshare-0.7.5-pyhd3eb1b0_1003
  prometheus_client  conda-forge::prometheus_client-0.11.0~ -->
pkgs/main::prometheus_client-0.11.0-pyhd3eb1b0_0
  prompt-toolkit     conda-forge::prompt-toolkit-3.0.21-py~ -->
pkgs/main::prompt-toolkit-3.0.20-pyhd3eb1b0_0
  pycparser          conda-forge::pycparser-2.20-pyh9f0ad1~ -->
pkgs/main::pycparser-2.20-py_2
  pygments           conda-forge::pygments-2.10.0-pyhd8ed1~ -->
```

```
pkgs/main::pygments-2.10.0-pyhd3eb1b0_0
  pyparsing          conda-forge::pyparsing-3.0.3-pyhd8ed1~ -->
pkgs/main::pyparsing-2.4.7-pyhd3eb1b0_0
  pysocks            conda-forge::pysocks-1.7.1-py39h6e949~ -->
pkgs/main::pysocks-1.7.1-py39hecd8cb5_0
  python-dateutil    conda-forge::python-dateutil-2.8.2-py~ -->
pkgs/main::python-dateutil-2.8.2-pyhd3eb1b0_0
  pytz                conda-forge::pytz-2021.3-pyhd8ed1ab_0 -->
pkgs/main::pytz-2021.3-pyhd3eb1b0_0
  pyzmq              conda-forge::pyzmq-22.3.0-py39h7fec2f~ -->
pkgs/main::pyzmq-22.2.1-py39h23ab428_1
  requests           conda-forge::requests-2.26.0-pyhd8ed1~ -->
pkgs/main::requests-2.26.0-pyhd3eb1b0_0
  six                  conda-forge::six-1.16.0-pyh6c4a22f_0 -->
pkgs/main::six-1.16.0-pyhd3eb1b0_0
  sniffio            conda-forge::sniffio-1.2.0-py39h6e949~ -->
pkgs/main::sniffio-1.2.0-py39hecd8cb5_1
  terminado          conda-forge::terminado-0.12.1-py39h6e~ -->
pkgs/main::terminado-0.9.4-py39hecd8cb5_0
  testpath           conda-forge::testpath-0.5.0-pyhd8ed1a~ -->
pkgs/main::testpath-0.5.0-pyhd3eb1b0_0
  tornado            conda-forge::tornado-6.1-py39h89e85a6~ -->
pkgs/main::tornado-6.1-py39h9ed2024_0
  traitlets          conda-forge::traitlets-5.1.1-pyhd8ed1~ -->
pkgs/main::traitlets-5.1.0-pyhd3eb1b0_0
  urllib3            conda-forge::urllib3-1.26.7-pyhd8ed1a~ -->
pkgs/main::urllib3-1.26.7-pyhd3eb1b0_0
  wcwidth            conda-forge::wcwidth-0.2.5-pyh9f0ad1d~ -->
pkgs/main::wcwidth-0.2.5-pyhd3eb1b0_0
  webencodings       conda-forge/noarch::webencodings-0.5.~ -->
pkgs/main/osx-64::webencodings-0.5.1-py39hecd8cb5_1
  zeromq               conda-forge::zeromq-4.3.4-he49afe7_1 -->
pkgs/main::zeromq-4.3.4-h23ab428_0
  zipp                 conda-forge::zipp-3.6.0-pyhd8ed1ab_0 -->
pkgs/main::zipp-3.6.0-pyhd3eb1b0_0


Downloading and Extracting Packages
brotlipy-0.7.0       | 333 KB    | ########## | 100%
anyio-2.2.0          | 125 KB    | ########## | 100%
notebook-6.4.5       | 4.2 MB    | ########## | 100%
cryptography-35.0.0  | 1.1 MB    | ########## | 100%
pysocks-1.7.1        | 31 KB     | ########## | 100%
jupyter_server-1.4.1 | 312 KB    | ########## | 100%
sniffio-1.2.0        | 15 KB     | ########## | 100%
Preparing transaction: …working… done
Verifying transaction: …working… done
```

```
Executing transaction: …working… done
```

[12]:
```
! python --version
```

```
Python 3.9.7
```

[18]:
```
# caching decorators are added in Python 3.9
from functools import cache

count = 0
@cache # same as @lru_cache(maxsize=None) # by default lru_cache store 128␣
↪entries
def cachedFib(n):
    global count
    count += 1
    if n <= 1:
        return 1
    f = fib(n-1) + fib(n-2)
    return f
```

[22]:
```
import timeit
n=20
print(timeit.timeit('cachedFib(n)', number=1, globals=globals()))
print('total call = ', count)
```

```
1.6439998944406398e-06
total call =  753945029
```

[24]:
```
# second time just looks up the cache as n = 20
count = 0
print(timeit.timeit('cachedFib(n)', number=1, globals=globals()))
print('total call = ', count)
```

```
1.1580000318645034e-06
total call =  0
```

[25]:
```
n = 30
count = 0
print(timeit.timeit('cachedFib(n)', number=1, globals=globals()))
print('total call = ', count)
```

```
0.25880884699995477
total call =  1664079
```

1.6.1 computational complexity of memoized fib

- Time Complexity - $O(n)$
- Space Complexity - $O(n)$

1.6.2 normally large integer answers are reported in mod

- mod of a farily large prime number e.g. $(10^9 + 7)$
- need to know some modular arithmetic: https://www.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/modular-addition-and-subtraction
- $(A + B)\%C = (A\%C + B\%C)\%C$
- $(A - B)\%C = (A\%C - B\%C)\%C$

[26]:
```
mod = 1000000007
def MemoizedModFib(memo, n):
    if n <= 2:
        return 1
    if n in memo:
        return memo[n]
    memo[n] = (MemoizedFib(memo, n-1)%mod + MemoizedFib(memo, n-2)%mod)%mod
    return memo[n]
```

[27]:
```
memo = {}
n=1000 #try 40, 50, 60, 100, 500, 10000, ...
print("fib at {}th position = {}".format(n, MemoizedModFib(memo, n)))
```

```
fib at 1000th position = 107579939
```

1.6.3 bottom-up (iterative) fibonacci solution

- first calculate fib(0) then fib(1), then fib(2), fib(3), and so on

[28]:
```
def iterativeFib(n):
    # fib array/list
    fib = [1]*(n+1) # initialize 0..n list with 1
    for i in range(2, n+1):
        fib[i] = fib[i-1] + fib[i-2]
    return fib[i]
```

[29]:
```
n=1000
print(timeit.timeit('iterativeFib(n)', number=1, globals=globals()))
# is faster than recursive counterpart
```

```
0.00025684899992484134
```

1.7 Coin Change Problem

- https://www.geeksforgeeks.org/understanding-the-coin-change-problem-with-dynamic-programming/
- essential to understanding the paradigm of DP
- a variation of problem definition:
  - Given an infinite number of coins of various denominations such as 1 cent (penny), 5 cents (nickel), and 10 cents (dime), can you determine the total number of combinations (order doesn't matter) of the coins in the given list to make up some amount N?

- Example 1:
  - Input: coins $= [1, 5, 10]$, $N = 8$
  - Output: 2
  - Combinations:
    * $1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 8$
    * $1 + 1 + 1 + 5 = 8 $
- Example 2:
  - Input: coins $= [1, 5, 10], N = 10$
  - Output: 4
  - Combinations:
    * $1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 10$
    * $ 1+1+1+1+1+5 = 10$
    * $ 5+5 = 10$
    * $10 = 10$

1.7.1 Algorithm

- we use tabulation/list/array to store the number of ways for outcome $N = 0$ to 12
  - bottom-up approach
- values of list represent the number of ways; indices represent the outcome/sum N
- so ways $= [0, 0, 0, 0, 0, 0, ...]$ initialized with N $0s$
- base case:
  - ways[0] = 1; there's 1 way to make sum $N = 0$ using 0 coin
- for each coin:
  - if the value of coin is less than the outcome/index n,
    * update the ways[n] = ways[n-coin] + ways[n]

[18]:
```python
def countWays(coins, N):
    # use ways table to store the results
    # ways[i] will store the number of solutions for value i
    ways = [0]*(N+1) # initialize all values 0-12 as 0
    # base case
    ways[0] = 1
    # pick all coins one by one
    # update the ways starting from the value of the picked coin
    print('indices/sum:', list(range(N+1)))
    for coin in coins:
        for n in range(N+1): # len(ways)
            if coin <= n:
                ways[n] += ways[n-coin]
        """better
        for n in range(coin, N+1):
            ways[n] += ways[n-coin]
        """
        print('ways:       ', ways, coin)
    return ways[N]
```

[15]:
```
coins = [1, 5, 10]
N = 8
print('Number of Ways to get {} = {}'.format(N, countWays(coins, N)))
```

```
indices/sum: [0, 1, 2, 3, 4, 5, 6, 7, 8]
ways:        [1, 1, 1, 1, 1, 1, 1, 1, 1] 1
ways:        [1, 1, 1, 1, 1, 2, 2, 2, 2] 5
ways:        [1, 1, 1, 1, 1, 2, 2, 2, 2] 10
Number of Ways to get 8 = 2
```

[16]:
```
N = 10
print('Number of Ways to get {} = {}'.format(N, countWays(coins, N)))
```

```
indices/sum: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
ways:        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] 1
ways:        [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3] 5
ways:        [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 4] 10
Number of Ways to get 10 = 4
```

[17]:
```
N = 12
print('Number of Ways to get {} = {}'.format(N, countWays(coins, N)))
```

```
indices/sum: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
ways:        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] 1
ways:        [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3] 5
ways:        [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 4, 4, 4] 10
Number of Ways to get 12 = 4
```

1.8 find minimum number of unique coins that make a given value/change

- Problem:
  - Input: $coins = [5, 10, 25], N = 30$
  - Output: 2
  - Combinations: $25 + 5 = 30$

[5]:
```
import math

# DP solution for min coin count to make the change N
def minCoins(coins, N):
    # count list stores the minimum number of coins required for i value
    # all values 0-N are initialized to infinity
    count = [math.inf]*(N+1)
    # base case
    # no. of coin required to make 0 value is 0
    count[0] = 0
    # computer min coins for all values from 1 to N
    for i in range(1, N+1):
        for coin in coins:
```

```
            # for every coin smaller than value i
            if coin <= i:
                if count[i-coin]+1 < count[i]:
                    count[i] = count[i-coin]+1
    return count[N]
```

[7]:
```
coins = [1, 3, 4]
N = 6
print('min unique coins required to give total of {} change = {}'.format(N,␣
 ↪minCoins(coins, N)))
```

```
min unique coins required to give total of 6 change = 2
```

1.9 Exercises

1. Ocean's Anti-11 - https://open.kattis.com/problems/anti11
   - Hint: count all possible n length binary integers (without 11) for the first few (2,3,4) positive integers and you'll see a Fibonaccii like pattern that gives the total number of possible binaries without 11 in them

- Write a program that finds factorials of a bunch of positive integers. Would memoization improve time complexity of the program?

[ ]:
```

```