In [4]:
```
def fib(n):
    if n == 0 or n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)
```

In [5]:
```
fib(0)
```

Out[5]: 1

In [6]:
```
fib(2)
```

Out[6]: 2

In [7]:
```
fib(3)
```

Out[7]: 3

In [8]:
```
fib(4)
```

Out[8]: 5

In [9]:
```
fib(10)
```

Out[9]: 89

In [10]:
```
fib(11)
```

Out[10]: 144

In [11]:
```
fib(20)
```

Out[11]: 10946

In [14]:
```
fib(30)
```

Out[14]: 1346269

In [15]:
```
fib(40)
```

Out[15]: 165580141

In [16]:
```
fib_cache = {}
def fib2(n):
    global fib_cache
    if n not in fib_cache:
        if n == 0 or n == 1:
            fib_cache[n] = 1
        else:
            fib_cache[n] = fib2(n-1) + fib2(n-2)
    return fib_cache[n]
```

In [17]:
```
fib2(0)
```

Out[17]:
```
1
```

In [18]:
```
fib2(1)
```

Out[18]:
```
1
```

In [19]:
```
fib2(30)
```

Out[19]:
```
1346269
```

In [20]:
```
fib2(40)
```

Out[20]:
```
165580141
```

In [21]:
```
fib2(100)
```

Out[21]:
```
573147844013817084101
```

In [22]:
```
fib2(1000)
```

Out[22]:
```
70330367711422815821835254877183549770181269836358732742604905087154
53711819693357974224949456261173348775044924176599108818636326545022
36471060120533741212738673391111981393731255987676900919022452453234
03501
```

In [25]:
```
fib_cache = {}
def fib2(n):
    global fib_cache
    if n not in fib_cache:
        if n == 0 or n == 1:
            fib_cache[n] = 1
        else:
            fib_cache[n] = fib2(n-1) + fib2(n-2)
        print(f"filling fib_cache[{n}]={fib_cache[n]}")
    return fib_cache[n]
```

In [26]:
```
fib2(6)
```

```
filling fib_cache[1]=1
filling fib_cache[0]=1
filling fib_cache[2]=2
filling fib_cache[3]=3
filling fib_cache[4]=5
filling fib_cache[5]=8
filling fib_cache[6]=13
```

Out[26]: 13

In [27]:
```
def fib3(n):
    fib_array = [None for _ in range(n+1)]
    fib_array[0] = 1
    fib_array[1] = 1
    for i in range(2,n+1):
        fib_array[i] = fib_array[i-1] + fib_array[i-2]
    return fib_array[n]
```

In [28]:
```
fib3(30)
```

Out[28]: 1346269

In [29]:
```
fib3(40)
```

Out[29]: 165580141

In [30]:
```
fib3(1000)
```

Out[30]:
```
70330367711422815821835254877183549770181269836358732742604905087154
53711819693357974224949456261173348775044924176599108818636326545022
36471060120533741212738673391111981393731255987676900919022452453234
03501
```

In [32]:
```
def segment(s, n=0):
    print(f"segment {s[n:]}")
    if n == len(s):
        return True
    for i in range(1, len(s)-n+1):
        if s[n:n+i] in words:
            if segment(s,n+i):
                return True
    return False
```

In [3]:
```
words = {"this","is","a","hard","course", "aha", "i", "a", "ah", "ha",
"his", "sis", "our","ours","har"}
```

In [35]:
```
segment("thisisahardcourse")
```

```
segment thisisahardcourse
segment isahardcourse
segment sahardcourse
segment ahardcourse
segment hardcourse
segment rdcourse
segment dcourse
segment course
segment
```

Out[35]: True

In [41]:
```
segment("thisisahardcourseq")
```

```
segment thisisahardcourseq
segment isahardcourseq
segment sahardcourseq
segment ahardcourseq
segment hardcourseq
segment rdcourseq
segment dcourseq
segment courseq
segment q
segment ardcourseq
segment rdcourseq
segment rdcourseq
```

Out[41]: False

In [10]:
```
segment_cache = {}
def segment(s, n=0):
    print(f"segment {s[n:]}")
    global segment_cache
    if (s,n) not in segment_cache:
        if n == len(s):
            segment_cache[(s,n)] = True
        else:
            for i in range(1, len(s)-n+1):
                if s[n:n+i] in words:
                    if segment(s,n+i):
                        segment_cache[(s,n)] = True
                        break
            if (s,n) not in segment_cache:
                segment_cache[(s,n)] = False
    else:
        print(".. cached")
    return segment_cache[(s,n)]
```

In [11]:
```
segment("thisisahardcourseq")
```

```
segment thisisahardcourseq
segment isahardcourseq
segment sahardcourseq
segment ahardcourseq
segment hardcourseq
segment rdcourseq
segment dcourseq
segment courseq
segment q
segment ardcourseq
segment rdcourseq
.. cached
segment rdcourseq
.. cached
```

Out[11]: False

In [12]:
```
def segment2(s):
    # segmentable[i] == s[i:] can be split into words
    segmentable = [None for _ in range(len(s)+1)]
    # s[n:n] = "" *is* segmentable
    n = len(s)
    segmentable[n] = True
    # n-1, ..., 0
    for i in range(n-1,-1,-1):
        for j in range(i+1,n+1):
            if s[i:j] in words and segmentable[j]:
                segmentable[i] = True
                break
        if segmentable[i] is None:
            segmentable[i] = False
    print(f"{segmentable}")
    return segmentable[0]
```

In [13]:
```
segment2("thisisahardcourse")
```

```
[True, True, True, True, True, False, True, True, False, False, Fals
e, True, False, False, False, False, False, True]
```

Out[13]: True