2.1 Understand

1) Fibonacci(20)

Let F[0...n] be a list of length n+1
F[0] = 0
F[1] = 1
For i = 2 to 20
  F[i] = F[i-1] + F[i-2]
EndFor
Return F[20]

2) $JS(n) = \begin{cases} 0 & \text{if } n=0 \\ P[1] & \text{if } n=1 \\ \max\begin{pmatrix} JS(n-2)+P[n], \\ JS(n-1) \end{pmatrix} & \text{if } n \geq 2 \end{cases}$

Step: Let J[0...n] be an array of size n+1
J[0] = 0
J[1] = P[1] // list starts from index 1
For i = 2 to n:
  a = J[i-2] + P[i]
  b = J[i-1]
  J[i] = max(a, b)
EndFor

Underline represents optimal answers for job set.

1) i=2, a=5 b=5 J[2]=5
2) i=3, a=5+12, b=5, J[3]=17
3) i=4, a=5+7, b=17, J[4]=17
4) i=5, a=17+5 b=17, J[5]=22
5) i=6, a=17+13, b=22, J[6]=30
6) i=7, a=22+7, b=30, J[7]=30
7) i=8 a=30+5, b=30, J[8]=35
8) i=9 a=30+4, b=35, J[9]=35
9) i=10 a=35+9, b=35, J[10]=44
10) i=11 a=35+8, b=44, J[11]=44
11) i=12 a=44+7, b=44, J[12]=51

i=13 a=44+5, b=51, J[13]=51
i=14 a=51+8, b=51, J[14]=59
i=15 a=51+4, b=59, J[15]=59
i=16 a=59+3, b=59, J[16]=62
i=17 a=59+5, b=62, J[17]=64
i=18 a=62+10, b=64, J[18]=72
i=19 a=64+4, b=72, J[19]=72
i=20 a=72+6, b=78, J[20]=78
i=21 a=72+8, b=78, J[21]=80
i=22 a=78+12, b=80, J[22]=90
i=23 a=80+5, b=90, J[23]=90
i=24 a=90+6, b=90, J[24]=96
i=25 a=90+3, b=96, J[25]=96
i=26 a=96+7, b=96, J[26]=103
i=27 a=96+16, b=103, J[27]=112
i=28 a=103+2, b=112, J[28]=112
i=29 a=112+2, b=112, J[29]=114
i=30 a=112+16, b=114, J[30]=128

J[30]=128 is the maximum earnings.
The optimal sol: {5,12,13,5,9,7,8,3,10,6,12,6,16,16}
The done wasted: {1,3,6,8,10,12,14,16,18,20,22,24,27,30}

3a) Input: Let A[1...n] be a list with n items where each index has a weight w.
Output: Let m represent the minimum number of cars required.

3b) The maximum number of cars required would be an n cars while the lower bound on the minimum is as follows: Let s be the sum of all weights in W[1...n] such that $s = \sum_{i=1}^{n} W[i]$, the minimum is $\lceil \frac{1}{1000} s \rceil$.

3c) Counterexample:
Let A[1...n] be a list w/ items of weight {510, 350, 490, 30, 620} where the index is set at 1.

* With this algorithm, if you're iterating through the list and placing an item in the first car it belongs to, you would require 3 cars while the minimum is 2 cars.
* For instance, on the 2nd iteration, you would place A[2] into the first car. A[3] does not fit into the 1st car, so you would place it into the 2nd car. A[4] would end up in the 2nd car, but A[5] doesn't fit, so would place it in the 3rd car.
* However, the optimal solution would be is A[1] and A[3] were in one car while A[2], A[4], and A[5] would be in another as those sums are ≤ 1000.
Thus, the first-fit algorithm doesn't hold.

3d) Counterexample:

Let A[1...n] be a list of items w/ weight {510, 350, 490, 30, 620} where the starting index is set to 1.

With the best-fit algorithm, A[1] would go into the first car. Then, A[2] would go into the first car as that is the car w/ the least amt. of free space. However, A[3] and A[4] would go into the 2nd car and A[5] would be forced to go into the 3rd car. As proven before, the minimum number of cars required is actually 2.

Thus, the best-fit algorithm does not result in the optimal solution.

2.2 Explore

1) Input: Let P[1...n] be a list with n payouts where each index represents a day of work.
Output: Let J[1...n] be an array that has the total, maximum earnings after each day.

2) $JS(n) = \begin{cases} 0 & \text{if } n=0 \\ P[1] & \text{if } n=1 \\ P[1]+P[2] & \text{if } n=2 \\ \max\begin{pmatrix} P[n]+P[n-1]+JS(n-3), \\ P[n]+JS(n-2), JS(n-1) \end{pmatrix} & \text{if } n \geq 3 \end{cases}$

at $n=3$, $\max(P[1]+P[2], P[2]+P[3])$
at $n=4$, $\max(P[1]+P[2]+P[4], P[1]+P[3]+P[4], P[2]+P[3])$
overlapping subproblems

3) JobSelection(P[1...n])
    Let J[0...n] be an array of n+1 length
    Let J[0] = 0
    Let J[1] = P[1]
    Let J[2] = P[1]+P[2]
    For int i=3 to n
        Let a = J[i-3]+P[i-1]+P[i]
        Let b = J[i-2]+P[i]
        Let c = J[i-1]
        J[i] = max(a,b,c)
    EndFor
    Return J[n]

4) SubSelection(P[1...n])

```
Let J[0...n] be an array of n+1 length
Let S[1...n] be an array of n length
Let J[0] = 0
Let J[1] = P[1]
Let J[2] = P[1]+P[2]
For int i=3 to n
  Let a = J[i-3] + P[i-1] + P[i]
  Let b = J[i-2] + P[i]
  Let c = J[i-1]
  J[i] = max(a,b,c)
  If J[i] = a, then S[i] = 2
  else if J[i] = b, then S[i] = 1
  else S[i] = 0
EndFor
```

```
Recover(S[1...n])
 Let R be an empty set.
 Let i = n
 while i > 0:
    if S[i] = 2 then,
       Add i to R
       let i = i-3
    else if S[i] = 1 then,
       Add i to R
       let i = i-2
    else
       let i = i-1
 EndWhile
 Return R
```

5) Steps:

i=3, a=9+12, b=5+12, c=5+9, J[3]=21

i=4, a=5+12+7, b=5+9+7, c=21, J[4]=24

set = {5, 12, 7}

i=5, a=5+9+7+5, b=21+5, c=24, J[5]=26

set = {5, 9, 7, 5}

i=6, a=21+5+13, b=24+13, c=26, J[6]=39

set = {9, 12, 5, 13}

i=7, a=24+13+7, b=26+7, c=39, J[7]=44

set = {5, 12, 7, 13, 7}

i=8, a=26+7+5, b=39+5, c=44, J[8]=44

set = {9, 12, 5, 13, 5}

i=9, a=39+5+4, b=44+4, c=44, J[9]=48

set = {9, 12, 5, 13, 5, 4}

i=10, a=44+4+9, b=44+9, c=48, J[10]=57

set = {9, 12, 5, 13, 5, 9}

i=11, a=44+9+8, b=48+8, c=57, J[11]=61

set = {9, 12, 5, 13, 5, 9, 8}

i=12, a=48+8+7, b=57+7, c=61, J[12]=64

i=13, a=57+7+5, b=61+5, c=64, J[13]=69

i=14, a=61+5+8, b=64+8, c=69, J[14]=74

set = {9, 12, 5, 13, 5, 9, 8, 5, 8}

i=15, a=64+8+9, b=69+4, c=74, J[15]=76

i=16, a=69+4+3, b=74+3, c=76, J[15]=77

set = {9, 12, 5, 13, 5, 9, 8, 5, 8, 3}

i=17, a=74+3+5, b=76+5, c=77, J[17]=82

set = {9, 12, 5, 13, 5, 9, 8, 5, 8, 3, 5}

i=18, a=76+5+10, b=77+10, c=84, J[18]=91

i=19, a=77+10+4, b=84+4, c=91, J[19]=91

set = {9, 12, 5, 13, 5, 9, 8, 5, 8, 3, 10, 4}

i=20, a=82+4+6, b=91+6, c=91, J[20]=97

i=21, a=91+6+8, b=97+8, c=97, J[21]=105

i = 22, a = 97+5+12, b = 97+12, c = 105, J[22] = 111
set = {9, 12, 5, 13, 5, 9, 8, 5, 8, 3, 10, 4, 8, 12}
i = 23, a = 97+12+5, b = 105+5, c = 113, J[23] = 114
i = 24, a = 105+5+6, b = 111+6, c = 114, J[22] = 117
set = {9, 12, 5, 13, 5, 9, 8, 5, 8, 3, 10, 9, 8, 12, 6}
i = 25, a = 111+6+3, b = 114+3, c = 117, J[25] = 120
set = {9, 12, 5, 13, 5, 9, 8, 5, 8, 3, 10, 4, 8, 12, 6, 3}
i = 26, a = 114+3+7, b = 117+7, c = 120, J[26] = 124
set = {9, 12, 5, 13, 5, 9, 8, 5, 8, 3, 10, 4, 8, 12, 6, 3, 7}
i = 27, a = 117+7+16, b = 120+16, c = 123, J[27] = 140
set = {9, 12, 5, 13, 5, 9, 8, 5, 8, 3, 10, 4, 8, 12, 6, 7, 16}
i = 28, a = 120+16+2, b = 124+2, c = 140, J[28] = 140
set = {9, 12, 5, 13, 5, 9, 8, 5, 8, 3, 10, 4, 8, 12, 6, 7, 16}
i = 29, a = 124+2+2, b = 140+2, c = 140, J[29] = 142
i = 30, a = 140+2+16, b = 140+16, c = 142, J[30] = 158
set = {9, 12, 5, 13, 5, 9, 8, 5, 8, 3, 10, 4, 8, 12, 6, 7, 16, 2, 16}
Max earnings is J[30] = 158,
Days worked {2, 3, 5, 6, 8, 10, 11, 13, 14, 16, 18, 19, 21, 22, 29, 26, 27, 29, 30}

(6) Time complexity is $\Theta(n)$ and space complexity is $\Theta(n)$.

2.3 Expand

1) Input: Let P[1...n] be a list of n prices per mile traveled, let D[1...n] be a list of n adjacent distances between two gas stations, and let m be the miles Bob can travel before each refill.

Output: Let M[1...n] be an array that holds the total minimum costs such that $M[i] = P[j] + \sum_{k=i+1}^{j} D[k] + M[i-1]$

2) Counter Example:

Suppose m=9. Let $D = \{1,2,3,4,1,2,3,4,1,2,3\}$ and let $P = \{5, 15, 25, 50, 30, 2, 8, 50, 12, 28, 3\}$. In this situation, Bob reaches the most costly gas station after traveling the maximum miles he can travel before each refill. This would incur the maximum cost rather than the minimum.

3) $MC(n,m) = \begin{cases} 0 & \text{if } n=0 \\ \min(P[j] \cdot d(i,j) + MC(i-1,m) & \text{if } n>0 \text{ and } d(i,j) <= m \text{ and } 0 <= i < j <= n \end{cases}$

```
Minimum(n, m)
  Let M[0...n] be an array of n+1 length
  Let M[0] = 0
  Let d = 0
  Let k = 0
  Let min = Integer.MAX_VALUE
  For i = 1 to n
    d = 0
    min = Integer.MAX_VALUE
    For j = i + 0
      d = d + D[j]
      if d > m, then break
      else
        if min > P[i]·d + M[j-1]
          min = P[i]·d + M[j-1]
    EndFor
    Let M[i] = min
  EndFor
  Return M[n]
```

5) $$\sum_{i=1}^{n}\left(c + \sum_{j=0}^{i} d\right) = \sum_{i=1}^{n}(c + d(i+1)) = \sum_{i=1}^{n} c + d\sum_{i=1}^{n} i + d\sum_{i=1}^{n} 1$$

$$= cn + \frac{dn(n+1)}{2} + dn = \Theta(n^2)$$

Worst-case runtime is $\Theta(n^2)$.

6) The minimum cost (at location n of M[1...n]) is M[n] = 10339

7) Counter Example:

Let m = 12, P = {12, 8, 4} and D = {4, 8, 4}

With the greedy algorithm, Bob visits gas station 2 and 3 which has a cost of 112. The optimal solution however, has Bob visit gas station 1 and 3 which has total cost of 96. This greedy algorithm

2.4 Challenge

1) The lower bound on the minimum number of cars is as follows,

Let s be the sum of all weights of each car in W[1...n] such that $s = \sum_{i=1}^{n} W[i]$. The minimum is $\lceil \frac{1}{1000} s \rceil$.

2) Proof: The first-fit algorithm will never leave more than one car less than half full because if an item's weight is over 500, it will automatically fulfill this requirement as the item's weight is over half of the car's maximum capacity to begin with. If the item's weight is under 500 or half, then lets assume we have two cars less than half full. This is a contradiction because if these items fit in the second car, they should've fit in the first car by default.

3) Proof: The best-fit algorithm will never leave more than one car less than half full. If an item's weight is over half, it will automatically fulfill this requirement. Else, it will always try to find a car whose current weight plus its own gets the car to as close as max capacity as possible. This is done recursively where items under 500 will attempt to get each car as close to max capacity as possible.

4) Suppose in the worst case, that every car is exactly half full except for the last one. Then the number of cars that would be needed in its worst case is

$$2s = 2\left\lceil \frac{1}{1000} \sum_{i=1}^{n} W[i] \right\rceil$$

This is at most double the number of cars needed in the optimal solution.