CS325-400

Homework 8

Sample Solutions

1) a)

First-Fit

Pseudocode:

```
FirstFit(items, C):
  bins = []
  numBins = 0

  bins[0] = new Bin
  bins[0].items += items[0]
  bins[0].weight += items[0].weight
  numBins += 1

  for i = 1 through items.length:
    packedItem = False
    for j = 0 through bins.length and not packedItem:
      if bins[j].weight + items[i].weight <= C:
        bins[j].items += items[i]
        bins[j].weight += items[i].weight
        packedItem = True

    if not packedItem:
      b = new bin
      b.items += items[i]
      b.weight += items[i].weight
      numBins += 1

  return numBins
```

Running Time:

Running time is $\mathbf{O(n^2)}$ because it's necessary to iterate through each item, and for each item iterate through all bags until the item is packed. In the worst case, each new item will not fit into any previously opened bag. Thus, the algorithm will first iterate through all opened bags, and then finally open a new bag and place the item. The number of bags will eventually reach *n,* and the number of total iterations will be $O(n^2)$.

First-Fit-Decreasing

Pseudocode:

```
FirstFitDecreasing(items, C):
  Sort items in decreasing order

  bins = []
  numBins = 0

  bins[0] = new Bin
  bins[0].items += items[0]
  bins[0].weight += items[0].weight
  numBins += 1

  for i = 1 through items.length:
    packedItem = False
    for j = 0 through bins.length and not packedItem:
```

```
      if bins[j].weight + items[i].weight <= C:
        bins[j].items += items[i]
        bins[j].weight += items[i].weight
        packedItem = True

    if not packedItem:
      b = new bin
      b.items += items[i]
      b.weight += items[i].weight
      numBins += 1

  return numBins
```

Running Time:

Running time is still $\mathbf{O(n^2)}$ for the reasons listed in First-Fit. In this case, we first perform a sort which is $O(n\lg n)$. However, the remainder of the algorithm is still the same and the algorithm will be dominated by the nested iteration, or $O(n^2)$. That said, as we saw in lecture the addition of sorting in this version of the algorithm can provide a more optimal result.

<u>Best Fit</u>

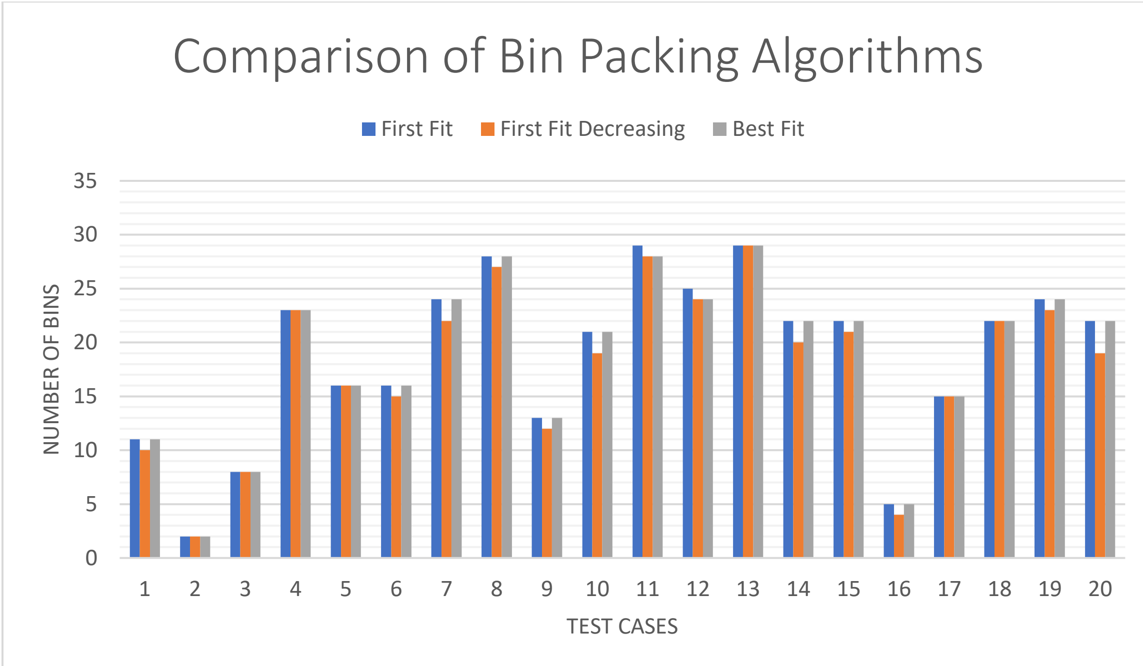
Pseudocode:

```
BestFit(items, C):
  bins = []
  numBins = 0

  bins[0] = new Bin
  bins[0].items += items[0]
  bins[0].weight += items[0].weight
  numBins += 1

  for i = 1 through items.length:
    bestfitIndex = -1
    minSpaceLeft = C + 1

    for j = 0 through bins.length:
      if bins[j].weight + items[i].weight <= C:
        if C - (bins[j].weight + items[i].weight) < minSpaceLeft:
          bestfitIndex = j
          minSpaceLeft = C - (bins[j].weight + items[i].weight)

    if bestfitIndex >= 0:
      bins[bestfitIndex].items += items[i]
      bins[bestFitIndex].weight += items[i].weight
    else:
      b = new bin
      b.items += items[i]
      b.weight += items[i].weight
      numBins += 1

  return numBins
```

Running Time:

Running time is again $\mathbf{O(n^2)}$. In this iteration, for each item we will always be iterating over all opened bins, and then either placing it in the best fit bin or a new bin if no fit was found. As mentioned in First-Fit, in the worst-case scenario the number of bins would approach n, and thus the nested loop will approach $O(n^2)$.

c) In 20 randomly generated bin-packing instances, for each test case all three of the algorithms gave similar numbers and when there was any difference in results it was relatively minimal—a difference between one and three bins. However, despite similar results, it was clear the **First Fit Decreasing** optimization provides a more optimal solution most often. In 11/20 test cases, First Fit Decreasing returned a smaller number of bins than either of the other algorithms. In 2/20 cases, both First Fit Decreasing and Best Fit returned the smallest value. In the remaining 7 cases, all three returned the same value.



I generated the file filled with test data for 20 random test cases using a python script. For each of the 20 cases, the script first outputs to file a random number between 5 and 100 representing the capacity of each bin for the case. Then, it generates a number between 3 and 50 of the number of items in the case. Finally, it generates random weights for each item in the test case between 1 and the capacity of the bins in that case. The file generated is in the format specified by the program, so it was able to be read into the program without any modifications.

2) **a) ANS:** 3 bins

Lindo Code

```
MIN y1 + y2 + y3 + y4 + y5 + y6
ST
    y1 + y2 + y3 + y4 + y5 + y6 >= 1

    4 x11 + 4 x12 + 4 x13 + 6 x14 + 6 x15 + 6 x16 - 10 y1 <= 0
    4 x21 + 4 x22 + 4 x23 + 6 x24 + 6 x25 + 6 x26 - 10 y2 <= 0
    4 x31 + 4 x32 + 4 x33 + 6 x34 + 6 x35 + 6 x36 - 10 y3 <= 0
    4 x41 + 4 x42 + 4 x43 + 6 x44 + 6 x45 + 6 x46 - 10 y4 <= 0
    4 x51 + 4 x52 + 4 x53 + 6 x54 + 6 x55 + 6 x56 - 10 y5 <= 0
    4 x61 + 4 x62 + 4 x63 + 6 x64 + 6 x65 + 6 x66 - 10 y6 <= 0

    x11 + x21 + x31 + x41 + x51 + x61 = 1
    x12 + x22 + x32 + x42 + x52 + x62 = 1
    x13 + x23 + x33 + x43 + x53 + x63 = 1
    x14 + x24 + x34 + x44 + x54 + x64 = 1
    x15 + x25 + x35 + x45 + x55 + x65 = 1
    x16 + x26 + x36 + x46 + x56 + x66 = 1
END
INT y1
INT y2
INT y3
INT y4
INT y5
```

```
INT y6
INT x11
INT x12
INT x13
INT x14
INT x15
INT x16
INT x21
INT x22
INT x23
INT x24
INT x25
INT x26
INT x31
INT x32
INT x33
INT x34
INT x35
INT x36
INT x41
INT x42
INT x43
INT x44
INT x45
INT x46
INT x51
INT x52
INT x53
INT x54
INT x55
INT x56
INT x61
INT x62
INT x63
INT x64
INT x65
INT x66
```

Output

```
LP OPTIMUM FOUND AT STEP     22
 OBJECTIVE VALUE =   3.00000000


 NEW INTEGER SOLUTION OF    3.00000000     AT BRANCH      0 PIVOT      22
 RE-INSTALLING BEST SOLUTION...

        OBJECTIVE FUNCTION VALUE

        1)      3.000000

  VARIABLE        VALUE          REDUCED COST
        Y1         0.000000          1.000000
        Y2         1.000000          1.000000
        Y3         0.000000          1.000000
        Y4         1.000000          1.000000
        Y5         1.000000          1.000000
        Y6         0.000000          1.000000
       X11         0.000000          0.000000
       X12         0.000000          0.000000
       X13         0.000000          0.000000
       X14         0.000000          0.000000
       X15         0.000000          0.000000
```

```
X16          0.000000           0.000000
X21          1.000000           0.000000
X22          0.000000           0.000000
X23          0.000000           0.000000
X24          0.000000           0.000000
X25          1.000000           0.000000
X26          0.000000           0.000000
X31          0.000000           0.000000
X32          0.000000           0.000000
X33          0.000000           0.000000
X34          0.000000           0.000000
X35          0.000000           0.000000
X36          0.000000           0.000000
X41          0.000000           0.000000
X42          1.000000           0.000000
X43          0.000000           0.000000
X44          1.000000           0.000000
X45          0.000000           0.000000
X46          0.000000           0.000000
X51          0.000000           0.000000
X52          0.000000           0.000000
X53          1.000000           0.000000
X54          0.000000           0.000000
X55          0.000000           0.000000
X56          1.000000           0.000000
X61          0.000000           0.000000
X62          0.000000           0.000000
X63          0.000000           0.000000
X64          0.000000           0.000000
X65          0.000000           0.000000
X66          0.000000           0.000000


ROW   SLACK OR SURPLUS     DUAL PRICES
 2)          2.000000           0.000000
 3)          0.000000           0.000000
 4)          0.000000           0.000000
 5)          0.000000           0.000000
 6)          0.000000           0.000000
 7)          0.000000           0.000000
 8)          0.000000           0.000000
 9)          0.000000           0.000000
10)          0.000000           0.000000
11)          0.000000           0.000000
12)          0.000000           0.000000
13)          0.000000           0.000000
14)          0.000000           0.000000

NO. ITERATIONS=      22
BRANCHES=    0 DETERM.=  1.000E    0
```

Interpretation: Bins 2, 4, and 5 were used (Y2, Y4, Y5). Items 1 and 5 were placed in bin 2 (X21, X25). Items 2 and 4 were placed in bin 4 (X42, X44). Items 3 and 6 were placed into bin 5 (X53, X56). The three bins were filled to capacity, so this must be the optimal value.

b) ANS: 3 bins

Lindo Code

```
MIN y1 + y2 + y3 + y4 + y5
ST
    y1 + y2 + y3 + y4 + y5 >= 1
```

```
    20 x11 + 10 x12 + 15 x13 + 10 x14 + 5 x15 - 20 y1 <= 0
    20 x21 + 10 x22 + 15 x23 + 10 x24 + 5 x25 - 20 y2 <= 0
    20 x31 + 10 x32 + 15 x33 + 10 x34 + 5 x35 - 20 y3 <= 0
    20 x41 + 10 x42 + 15 x43 + 10 x44 + 5 x45 - 20 y4 <= 0
    20 x51 + 10 x52 + 15 x53 + 10 x54 + 5 x55 - 20 y5 <= 0

    x11 + x21 + x31 + x41 + x51 = 1
    x12 + x22 + x32 + x42 + x52 = 1
    x13 + x23 + x33 + x43 + x53 = 1
    x14 + x24 + x34 + x44 + x54 = 1
    x15 + x25 + x35 + x45 + x55 = 1
END
INT y1
INT y2
INT y3
INT y4
INT y5
INT x11
INT x12
INT x13
INT x14
INT x15
INT x21
INT x22
INT x23
INT x24
INT x25
INT x31
INT x32
INT x33
INT x34
INT x35
INT x41
INT x42
INT x43
INT x44
INT x45
INT x51
INT x52
INT x53
INT x54
INT x55
```

Output

```
LP OPTIMUM FOUND AT STEP     17
 OBJECTIVE VALUE =   3.00000000


 NEW INTEGER SOLUTION OF    3.00000000     AT BRANCH      0 PIVOT      17
 RE-INSTALLING BEST SOLUTION...

        OBJECTIVE FUNCTION VALUE

        1)      3.000000

  VARIABLE        VALUE          REDUCED COST
        Y1         1.000000          1.000000
        Y2         1.000000          1.000000
        Y3         1.000000          1.000000
        Y4         0.000000          1.000000
        Y5         0.000000          1.000000
       X11         0.000000          0.000000
```

```
       X12          0.000000           0.000000
       X13          1.000000           0.000000
       X14          0.000000           0.000000
       X15          1.000000           0.000000
       X21          1.000000           0.000000
       X22          0.000000           0.000000
       X23          0.000000           0.000000
       X24          0.000000           0.000000
       X25          0.000000           0.000000
       X31          0.000000           0.000000
       X32          1.000000           0.000000
       X33          0.000000           0.000000
       X34          1.000000           0.000000
       X35          0.000000           0.000000
       X41          0.000000           0.000000
       X42          0.000000           0.000000
       X43          0.000000           0.000000
       X44          0.000000           0.000000
       X45          0.000000           0.000000
       X51          0.000000           0.000000
       X52          0.000000           0.000000
       X53          0.000000           0.000000
       X54          0.000000           0.000000
       X55          0.000000           0.000000


       ROW   SLACK OR SURPLUS     DUAL PRICES
        2)          2.000000           0.000000
        3)          0.000000           0.000000
        4)          0.000000           0.000000
        5)          0.000000           0.000000
        6)          0.000000           0.000000
        7)          0.000000           0.000000
        8)          0.000000           0.000000
        9)          0.000000           0.000000
       10)          0.000000           0.000000
       11)          0.000000           0.000000
       12)          0.000000           0.000000

 NO. ITERATIONS=      17
 BRANCHES=    0 DETERM.=  1.000E    0
```

Interpretation: Bins 1, 2, and 3 were used (Y1, Y2, Y3). Items 3 and 5 were placed in bin 1 (X13, X15) with a weight of 15 + 5 = 20. Item 1 was placed into bin 2 (X21) with a weight of 20. Items 2 and 4 were placed into bin 3 (X32, X34) with a weight of 10 + 10 = 20. The three bins were filled to capacity, so this must be the optimal value.