Xiaoying Li
CS 325 -Winter 2020
Homework #8

- **Problem 1**

a.

(1) First-Fit

```
int First-Fit (int weight[], int number, int capacity) {
    // Initialize bin's number to 0.
    int binNumber = 0;
    // Allocate an array to store the left room in bins, whose size is the number of items.
    int* binRoom;
    binRoom = new int[number];

    // Pack items one by one.
    for (int i = 0; i < number; i++) {
        int j;
        // Put each item into the first bin which it fits.
        for (j = 0; j < binNumber; j++) {
            // If such a bin is found, update this bin's left room and move to next item.
            if (binRoom[j] >= weight[i]) {
                binRoom[j] = binRoom[j] - weight[i];
                break;
            }
        }
        // If there is no available bin then open a new bin, set its left room,
        // and update bins' number, then move to the next item.
        if (j == binNumber) {
            binRoom[binNumber] = capacity - weight[i];
            binNumber++;
        }
    }

    delete[] binRoom;
    // After all items are packed into bins, return the number of bins.
    return binNumber;
}
```

The outer loop runs n times, the inner loop runs up to n times. Therefore, the running time of First-Fit algorithm is $O(n^2)$, where n is the number of items.

(2) First-Fit-Decreasing

```
int First-Fit-Decreasing(int weight[], int number, int capacity) {
	// First sort the items in decreasing order by size.
	Insertion-Sort(weight, number);
	// Then use First-Fit on the resulting list and return the number of bins.
	return First-Fit(weight, number, capacity);
}
```

I used insertion sort to sort the items, and its running time is $O(n^2)$; the running time of First-Fit is also $O(n^2)$. Therefore, the running time of First-Fit-Decreasing algorithm is $O(n^2)$, where n is the number of items.

(3) Best-Fit

```
int Best-Fit(int weight[], int number, int capacity) {
	// Initialize bin's number to 0.
	int binNumber = 0;
	// Allocate an array to store the left room in bins, whose size is the number of items.
	int* binRoom;
	binRoom = new int[number];

	// Pack items one by one.
	for (int i = 0; i < number; i++) {
		int j;
		// Initialize the least left room to bin's capacity plus 1,
		// and the index of bin with the least left room to 0;
		int least = capacity + 1;
		int index = 0;

		// Place the item into the bin which will leave the least room left over after the
		// item is placed in the bin.
		for (j = 0; j < binNumber; j++) {
			// Compare every bin's left room after the item is placed in the bin if the
			// item fits the bin to find the bin which will leave the least room left over
			// after the item is placed in the bin.
			if (binRoom[j] >= weight[i] && binRoom[j] - weight[i] < least) {
				// Update the least left room and the index of the bin with the least
				// left room.
				index = j;
				least = binRoom[j] - weight[i];
			}
		}
```

```
        // If the item does not fit in any bin, start a new bin, set its left room, and
        // update bins' number, then move to the next item.
        if (least == capacity + 1) {
            binRoom[binNumber] = capacity - weight[i];
            binNumber++;
        }

        // If such a bin is found, update this bin's left room and move to the next item.
        else {
            binRoom[index] -= weight[i];
        }
    }

    delete[] binRoom;
    // After all items are packed into bins, return the number of bins.
    return binNumber;
}
```

The outer loop runs n times, the inner loop runs up to n times. Therefore, the running time of Best-Fit algorithm is $O(n^2)$, where n is the number of items.

b. The answer is submitted to TEACH.

c.

Below are the screen shots of my randomly generated cases. In my code to produce the random inputs, the range of bin's capacity and number of items are both [1, 50], and the weight of each item is set to from 1 to not exceed the capacity of a bin. Totally 20 bin packing instances are generated.

```
Test Case 1:
    capacity: 45
    item number: 36
    weight: 20 43 21 43 11 30 1 43 41 21 35 28 15 22 40 25 12 5 36 41 38 43 10 38 45 3 40 35 43 30 41 29 37 10 40 32
    result: First Fit - 27; First Fit Decreasing - 26; Best Fit - 26

Test Case 2:
    capacity: 21
    item number: 13
    weight: 13 6 12 15 20 6 13 7 11 21 4 20 12
    result: First Fit - 9; First Fit Decreasing - 9; Best Fit - 9

Test Case 3:
    capacity: 35
    item number: 38
    weight: 26 24 1 19 19 8 10 20 3 26 14 11 2 35 11 16 19 13 33 2 16 7 34 21 20 21 20 4 16 19 3 9 8 34 2 24 23 26
    result: First Fit - 20; First Fit Decreasing - 19; Best Fit - 20

Test Case 4:
    capacity: 26
    item number: 9
    weight: 23 8 19 26 21 19 16 5 2
    result: First Fit - 6; First Fit Decreasing - 6; Best Fit - 6

Test Case 5:
    capacity: 35
    item number: 8
    weight: 30 19 25 4 3 15 32 15
    result: First Fit - 5; First Fit Decreasing - 5; Best Fit - 5

Test Case 6:
    capacity: 45
    item number: 9
    weight: 18 1 4 17 38 42 14 8 15
    result: First Fit - 4; First Fit Decreasing - 4; Best Fit - 4

Test Case 7:
    capacity: 11
    item number: 45
    weight: 10 7 5 6 7 8 1 7 1 6 11 2 1 11 10 8 8 1 7 8 9 8 6 10 8 7 3 4 1 8 9 10 5 11 9 6 1 10 8 5 3 8 2 8 10
    result: First Fit - 31; First Fit Decreasing - 31; Best Fit - 31

Test Case 8:
    capacity: 29
    item number: 47
    weight: 5 27 19 8 13 19 8 14 22 18 14 7 4 1 23 4 12 28 17 5 6 26 28 8 12 16 24 7 29 1 29 12 29 4 28 15 22 3 11 5 5 7 13 8 27 28 10
    result: First Fit - 25; First Fit Decreasing - 24; Best Fit - 25
```

```
Test Case 9:
    capacity: 48
    item number: 20
    weight: 33 17 6 21 35 27 12 15 47 35 31 9 16 11 24 19 47 19 13 29
    result: First Fit - 11; First Fit Decreasing - 10; Best Fit - 11

Test Case 10:
    capacity: 23
    item number: 34
    weight: 14 1 18 23 8 16 19 21 23 10 8 3 6 16 15 13 3 2 17 2 16 8 19 8 21 19 7 14 20 8 20 5 22 10
    result: First Fit - 21; First Fit Decreasing - 21; Best Fit - 21

Test Case 11:
    capacity: 3
    item number: 35
    weight: 3 1 3 2 3 2 1 1 3 2 2 1 2 2 3 1 3 1 2 1 3 2 1 2 3 2 2 3 3 3 2 3 1 3 1
    result: First Fit - 25; First Fit Decreasing - 25; Best Fit - 25

Test Case 12:
    capacity: 3
    item number: 44
    weight: 2 1 2 3 1 1 1 2 2 1 2 2 1 1 3 2 2 1 2 1 1 1 3 3 3 3 3 3 3 2 2 3 1 3 2 1 1 1 3 3 2 2 1 2
    result: First Fit - 29; First Fit Decreasing - 29; Best Fit - 29

Test Case 13:
    capacity: 48
    item number: 5
    weight: 10 6 27 16 3
    result: First Fit - 2; First Fit Decreasing - 2; Best Fit - 2

Test Case 14:
    capacity: 40
    item number: 9
    weight: 20 11 20 28 37 30 25 34 4
    result: First Fit - 7; First Fit Decreasing - 6; Best Fit - 7

Test Case 15:
    capacity: 15
    item number: 44
    weight: 15 12 14 4 4 7 6 9 5 7 11 4 10 1 15 8 3 2 6 5 13 3 10 12 1 3 3 4 5 14 2 13 15 1 11 10 5 11 8 3 1 11 3 14
    result: First Fit - 23; First Fit Decreasing - 22; Best Fit - 23

Test Case 16:
    capacity: 46
    item number: 26
    weight: 13 39 46 21 2 45 18 23 36 38 41 37 10 2 10 3 31 31 20 28 2 36 18 24 10 11
    result: First Fit - 15; First Fit Decreasing - 14; Best Fit - 14
```

```
Test Case 17:
    capacity: 45
    item number: 49
    weight: 32 30 20 23 3 39 33 38 26 3 16 7 31 24 31 35 32 40 5 40 9 10 14 22 15 29 6 41 43 32 35 15 18 35 44 1 27 35 24 22 28 19 20 45 24 14 27 21 19
    result: First Fit - 31; First Fit Decreasing - 29; Best Fit - 30

Test Case 18:
    capacity: 34
    item number: 46
    weight: 22 28 18 33 20 20 31 19 32 31 7 19 5 15 33 16 19 14 21 31 17 7 6 23 32 20 3 28 11 23 14 32 27 21 22 26 8 1 23 10 6 24 15 28 2 20
    result: First Fit - 31; First Fit Decreasing - 30; Best Fit - 30

Test Case 19:
    capacity: 18
    item number: 35
    weight: 11 5 16 2 1 11 6 9 11 7 10 16 10 5 8 13 11 6 17 4 18 4 5 13 4 10 13 4 13 18 15 6 15 7 11
    result: First Fit - 21; First Fit Decreasing - 20; Best Fit - 20

Test Case 20:
    capacity: 12
    item number: 46
    weight: 10 12 3 2 5 12 4 11 1 6 3 6 11 7 6 3 8 8 2 4 10 4 6 12 10 2 9 10 2 9 7 8 5 5 1 8 9 7 2 12 8 8 8 5 9 10
    result: First Fit - 29; First Fit Decreasing - 28; Best Fit - 29
```

The results for each algorithm are summarized as below, the algorithm with best performance is marked in blue for every test case.

Case Number	First-Fit	First-Fit- Decreasing	Best-Fit
1	27	26	26
2	9	9	9
3	20	19	20
4	6	6	6
5	5	5	5
6	4	4	4
7	31	31	31
8	25	24	25
9	11	10	11
10	21	21	21
11	25	25	25
12	29	29	29
13	2	2	2
14	7	6	7
15	23	22	23
16	15	14	14
17	31	29	30
18	31	30	30
19	21	20	20
20	29	28	29

Obviously, First-Fit-Decreasing algorithm performs best since it performs best in 11 cases over 20 test cases. Best-Fit algorithm performs as well as First-Fit-Decreasing in 4 cases, but was never better than First-Fit-Decreasing. And First-Fit algorithm performs worst, which never performs better than the other two algorithms. But there are 9 cases over 20 test cases where the three algorithm's performance is same.

Therefore, First-Fit-Decreasing algorithm performs better, in 11 cases over 20 test cases (55% times).

- **Problem 2**

a.

Code and result of the integer program from LINDO:

```
MIN Y1 + Y2 + Y3 + Y4 + Y5 + Y6
ST
        A1 + A2 + A3 + A4 + A5 + A6 = 1
        B1 + B2 + B3 + B4 + B5 + B6 = 1
        C1 + C2 + C3 + C4 + C5 + C6 = 1
        D1 + D2 + D3 + D4 + D5 + D6 = 1
        E1 + E2 + E3 + E4 + E5 + E6 = 1
        F1 + F2 + F3 + F4 + F5 + F6 = 1
        4A1 + 4B1 + 4C1 + 6D1 + 6E1 + 6F1 - 10Y1 <= 0
        4A2 + 4B2 + 4C2 + 6D2 + 6E2 + 6F2 - 10Y2 <= 0
        4A3 + 4B3 + 4C3 + 6D3 + 6E3 + 6F3 - 10Y3 <= 0
        4A4 + 4B4 + 4C4 + 6D4 + 6E4 + 6F4 - 10Y4 <= 0
        4A5 + 4B5 + 4C5 + 6D5 + 6E5 + 6F5 - 10Y5 <= 0
        4A6 + 4B6 + 4C6 + 6D6 + 6E6 + 6F6 - 10Y6 <= 0
END
        INT Y1
        INT Y2
        INT Y3
        INT Y4
        INT Y5
        INT Y6
        INT A1
        INT A2
        INT A3
        INT A4
        INT A5
        INT A6
        INT B1
        INT B2
        INT B3
        INT B4
        INT B5
        INT B6
        INT C1
        INT C2
        INT C3
        INT C4
        INT C5
        INT C6
        INT D1
        INT D2
        INT D3
        INT D4
        INT D5
        INT D6
        INT E1
        INT E2
        INT E3
        INT E4
        INT E5
        INT E6
        INT F1
        INT F2
        INT F3
        INT F4
        INT F5
        INT F6
```

```
LP OPTIMUM FOUND AT STEP     55
OBJECTIVE VALUE =   3.00000000


NEW INTEGER SOLUTION OF    3.00000000     AT BRANCH      0 PIVOT      55
RE-INSTALLING BEST SOLUTION...

    OBJECTIVE FUNCTION VALUE

    1)      3.000000

 VARIABLE        VALUE          REDUCED COST
       Y1         1.000000          1.000000
       Y2         0.000000          1.000000
       Y3         0.000000          1.000000
       Y4         0.000000          1.000000
       Y5         1.000000          1.000000
       Y6         1.000000          1.000000
       A1         0.000000          0.000000
       A2         0.000000          0.000000
       A3         0.000000          0.000000
       A4         0.000000          0.000000
       A5         1.000000          0.000000
       A6         0.000000          0.000000
       B1         1.000000          0.000000
       B2         0.000000          0.000000
       B3         0.000000          0.000000
       B4         0.000000          0.000000
       B5         0.000000          0.000000
       B6         0.000000          0.000000
       C1         0.000000          0.000000
       C2         0.000000          0.000000
       C3         0.000000          0.000000
       C4         0.000000          0.000000
       C5         0.000000          0.000000
       C6         1.000000          0.000000
       D1         0.000000          0.000000
       D2         0.000000          0.000000
       D3         0.000000          0.000000
       D4         0.000000          0.000000
       D5         1.000000          0.000000
       D6         0.000000          0.000000
       E1         0.000000          0.000000
       E2         0.000000          0.000000
       E3         0.000000          0.000000
       E4         0.000000          0.000000
       E5         0.000000          0.000000
       E6         1.000000          0.000000
       F1         1.000000          0.000000
       F2         0.000000          0.000000
       F3         0.000000          0.000000
       F4         0.000000          0.000000
       F5         0.000000          0.000000
       F6         0.000000          0.000000


      ROW   SLACK OR SURPLUS     DUAL PRICES
       2)         0.000000          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

NO. ITERATIONS=      55
BRANCHES=    0 DETERM.=  1.000E    0
```

Interpretation:

The result shows that Y1=1, Y5=1, Y6=1, A5=1, B1=1, C6=1, D5=1, E6=1, and F1=1, so the 6 items can be packed into at least 3 bins:

Y1 – B1 and F1: first bin - items of weight 4 and 6;

Y5 – A5 and D5: second bin – items of weight 4 and 6;

Y6 – C6 and E6: third bin – items of weight 4 and 6.

b.

Code and result of the integer program from LINDO:

```
MIN Y1 + Y2 + Y3 + Y4 + Y5
ST
        A1 + A2 + A3 + A4 + A5 = 1
        B1 + B2 + B3 + B4 + B5 = 1
        C1 + C2 + C3 + C4 + C5 = 1
        D1 + D2 + D3 + D4 + D5 = 1
        E1 + E2 + E3 + E4 + E5 = 1
        20A1 + 10B1 + 15C1 + 10D1 + 5E1 - 20Y1 <= 0
        20A2 + 10B2 + 15C2 + 10D2 + 5E2 - 20Y2 <= 0
        20A3 + 10B3 + 15C3 + 10D3 + 5E3 - 20Y3 <= 0
        20A4 + 10B4 + 15C4 + 10D4 + 5E4 - 20Y4 <= 0
        20A5 + 10B5 + 15C5 + 10D5 + 5E5 - 20Y5 <= 0
END
        INT Y1
        INT Y2
        INT Y3
        INT Y4
        INT Y5
        INT A1
        INT A2
        INT A3
        INT A4
        INT A5
        INT B1
        INT B2
        INT B3
        INT B4
        INT B5
        INT C1
        INT C2
        INT C3
        INT C4
        INT C5
        INT D1
        INT D2
        INT D3
        INT D4
        INT D5
        INT E1
        INT E2
        INT E3
        INT E4
        INT E5
```

```
LP OPTIMUM FOUND AT STEP     19
OBJECTIVE VALUE =   3.00000000


NEW INTEGER SOLUTION OF    3.00000000    AT BRANCH     0 PIVOT     19
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
       A1         0.000000          0.000000
       A2         1.000000          0.000000
       A3         0.000000          0.000000
       A4         0.000000          0.000000
       A5         0.000000          0.000000
       B1         1.000000          0.000000
       B2         0.000000          0.000000
       B3         0.000000          0.000000
       B4         0.000000          0.000000
       B5         0.000000          0.000000
       C1         0.000000          0.000000
       C2         0.000000          0.000000
       C3         1.000000          0.000000
       C4         0.000000          0.000000
       C5         0.000000          0.000000
       D1         1.000000          0.000000
       D2         0.000000          0.000000
       D3         0.000000          0.000000
       D4         0.000000          0.000000
       D5         0.000000          0.000000
       E1         0.000000          0.000000
       E2         0.000000          0.000000
       E3         1.000000          0.000000
       E4         0.000000          0.000000
       E5         0.000000          0.000000


      ROW   SLACK OR SURPLUS     DUAL PRICES
       2)         0.000000          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

NO. ITERATIONS=      19
BRANCHES=    0 DETERM.=  1.000E    0
```

Interpretation:

The result shows that Y1=1, Y2=1, Y3=1, A2=1, B1=1, C3=1, D1=1, and E3=1, so the 5 items can be packed into at least 3 bins:

Y1 – B1 and D1: first bin - items of weight 10 and 10;

Y2 – A2: second bin – item of weight 20;

Y3 – C3 and E3: third bin – items of weight 15 and 5.