**Artificial and Computational Intelligence**

**Assignment 3**

**Packet delivery agent**

**Problem statement**

Assume that you are in charge of an online e-commerce site which collects goods from vendors, stores them in godowns and later deliver them based on order placed. You have been assigned with robots that pick up packets from the front office and stores them in appropriate rooms. The packets are of different weights. The robot can carry 10 Kgs at a time. As in charge of the facility, you are required to give specific instructions to the robot to carry packets and store them in the rooms which are of different capacities.

The problem here is to find the correct combination of packets that can be carried by the robot so as to minimize the number of commutes by the robot to complete the job, the number of rooms used by the robot to store the contingency, and the remaining storage space.

The following figure gives the environment of the robot:

20 Kg

28 Kg

5 Kg

Pathway

**Agent’s Direction**

**Agent’s Direction**

46 Kg

35 Kg

25 Kg

Pathway

Front Office

The below table gives the Packet weights and number of packets:

|  |  |  |
| --- | --- | --- |
| Packet weight (Kg) | No. of packets | Total Weight (Kg) |
| 1 | 15 | 15 |
| 2 | 5 | 10 |
| 4 | 7 | 28 |
| 5 | 2 | 10 |
| 6 | 3 | 18 |
| 7 | 2 | 14 |
| 9 | 4 | 36 |
| Total 38 packets 131 Kgs | | |

1. Explain the heuristic that can be used to solve the problem? Justify your choice.
2. Explain the cost function associated with your search in reaching the goal.
3. Choose the correct algorithm suitable for this grid search.
4. Implement the algorithm in PYTHON
5. Print the number of commutes by the robot, the weight carried by the robot in each commute, the number of rooms used to store the contingency and the remaining space for the store keeper’s reference.

The assignment will be evaluated on the following points

1. Explanation of the heuristic and algorithm chosen [20% weightage]
2. Representation of the environment, fringe and the data structures used [20% weightage]
3. Implementation of the algorithm in PYTHON [40% weightage]
4. The details given by the algorithm to the store keeper. [20% weightage]

Note:

1. Kindly avoid plagiarism
2. Use suitable data structures to represent the environment, the solution space and the solution.
3. Python template is provided for your reference.

Hint:

Represent the number of commutes, the weight carried by the robot at each commute and the available room capacity as an objective vector. Use these features to define your evaluation function. Find the optimum combination to reach the goal of storing the contingency with minimum number of commutes and less number of rooms.

**Solution:**

1. **Explain the heuristic that can be used to solve the problem? Justify your choice.**

* **States**: A state specifies the number of packages still awaiting to be stored and available room spaces.
* **Initial state**: All rooms are empty\available i.e. Available Space = Initial Capacity, termed as “Closed State”
* **Actions:** To move Packages having total weight <= 10 KG into available space using a robot in each commute.
* **Transition Model**: Store package <= 10 KG from available packages into available rooms, providing a new transition state to Goal State.
* **Goal Test:** Checks whether all packages have been moved from front office to rooms.
* **Step Cost:** Fixed cost “1” unit is assigned for any action.
* **Path Cost:** Total “**Step Cost”** involved in storing packages in single commute and total commute of robot (journey from front office and back to front office, 2 units are assigned for this action)

**PEAS analysis for given robot problem:**

*Performance:* Move all packets from front office to rooms using minimum number of commute and minimum number of rooms with minimum free space available in the selected rooms.

*Environment:*  We have 6 rooms with max available space to hold - 46, 35, 25, 20, 28, 5 kgs. We have a total of 38 packets of 131 kgs and two paths for a robot to carry maximum of 10 kgs weight.

*Actuators:* Robot delivers (action) the packet to the room based on the available space in that particular room.

*Sensor*: Agent senses the number of available spaces.

On further analysis of given problem statement, we can relate it to Container/Bin Loading Problem. Container loading problems mainly address the issues of planning the loading order and loading position on the basis of ensuring certain constraints. In this particular scenario is weight the robot can carry. The container loading problem is a Nondeterministic Polynomial- (NP-) Hard problem.

For selecting the package, we have to carry maximum possible load of 10kgs on robot for each commute. For placing the selected weight in the room, we have to check if the weight of the package is less than or equal to available free space in the selected room.

1. **Explain the cost function associated with your search in reaching the goal.**

***Objective:***

1. To minimize the number of commutes of robot to complete the job.
2. To minimize the number of rooms to store the packets.
3. To maximize the space utilization in any selected room.

***Optimum Goal State:***

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Iteration No** | **Weights** | **46 - Block** | **35 Block** | **28 Block** | **25 Block** | **20 Block** | **5 Block** | **Steps** |  |
| 1 | 9+1 | 36 | 35 | 28 | 25 | 20 | 5 | 1+1 | 2 |
| 2 | 9+1 | 26 | 35 | 28 | 25 | 20 | 5 | 1+1 | 2 |
| 3 | 9+1 | 16 | 35 | 28 | 25 | 20 | 5 | 1+1 | 2 |
| 4 | 9+1 | 6 | 35 | 28 | 25 | 20 | 5 | 1 + 1 | 2 |
| 5 | 7+2+1 | 3 | 28 | 28 | 25 | 20 | 5 | 1+1+1 | 3 |
| 6 | 7+2+1 | 0 | 21 | 28 | 25 | 20 | 5 | 1 + 1 + 1 | 3 |
| 7 | 6+4 | 0 | 11 | 28 | 25 | 20 | 5 | 1+1 | 2 |
| 8 | 6+4 | 0 | 1 | 28 | 25 | 20 | 5 | 1 + 1 | 2 |
| 9 | 6+4 | 0 | 1 | 18 | 25 | 20 | 5 | 1+1 | 2 |
| 10 | 5+5 | 0 | 1 | 8 | 25 | 20 | 5 | 1+1 | 2 |
| 11 | 4+4+2 | 0 | 1 | 0 | 23 | 20 | 5 | 1+1+1 | 3 |
| 12 | 4+4+2 | 0 | 1 | 0 | 13 | 20 | 5 | 1 + 1 | 2 |
| 13 | 2+1+1+1+1+1+1+1 | 0 | 0 | 0 | 4 | 20 | 5 | 1 + 1 +1 | 3 |
| 14 | 1 | 0 | 0 | 0 | 3 | 20 | 5 | 1+1 | 2 |
|  |  | Remaining Weights after each iteration | | | | | | Total Steps | 32 |

There is limit of 10 KG weight that robot can carry and both room space and package weights are of varied capacities. Hence to optimize the number of commutes by the robot ensuring using minimum rooms space, we may have to store weights in multiple rooms in single commute adding multiple steps cost. This approach is used to design the heuristic function for problem.

We cannot use the Distance Calculation from Front Office to Room Location as sufficient location\measurement data is not provided hence to simplify the problem and assuming no constraint we used Step Cost of “1” to store package in a room and Path Cost will include all Step Cost incurred in single commute.

For example:

1. Consider (9 ,1) packages to be placed in 46-Block, where available free space is 46.

*Step 1: Move from Front office to 46-Block: 1 unit*

*Step 2: Move from 46-Block back to Front office: 1 unit*

***Total cost for this commute is 2***

1. Consider (7,2,1) packages to be placed in 46-Block and 35-Block. Available free space in 46 Block is 6 Kgs and in 35-block is 35 Kgs.

*Step 1: Move from Front office to 46-Block, place 3 kgs weight: 1 unit*

*Step 2: Move from 46-Block to 35-Block, place 7 kgs: 1 unit*

*Step 3: Move from 35-Block back to Front office: 1 unit*

***Total cost for this commute is 3***

1. **Choose the correct algorithm suitable for this grid search.**

We have used branch-and-bound algorithm for selecting the package and for placing the selected package in rooms we have evaluated with First Fit and Best-Fit algorithm.

For weight and bin selection, we have used ***branch-and-bound*** algorithm. In this algorithm, we combine max available weight and keep min weights to ensure we load robot with maximum of 10 kgs package. For the given problem, selected weight combinates for different iterations are:

|  |  |
| --- | --- |
| **Iteration No** | **Weights** |
| 1 | 9+1 |
| 2 | 9+1 |
| 3 | 9+1 |
| 4 | 9+1 |
| 5 | 7+2+1 |
| 6 | 7+2+1 |
| 7 | 6+4 |
| 8 | 6+4 |
| 9 | 6+4 |
| 10 | 5+5 |
| 11 | 4+4+2 |
| 12 | 4+4+2 |
| 13 | 2+1+1+1+1+1+1+1 |
| 14 | 1 |

1. *We have evaluated two algorithms for selecting the weights and placing the selected weights in the room based on free space available:*
2. ***First Fit Bin Loading***

* Scan the bins in order and place the new item in the first bin that is large enough to hold it. A new bin is created only when an item does not fit in the previous bins.
* The above implementation of First Fit requires O(n2) time, but First Fit can be implemented in O(n Log n) time using Self-Balancing Binary Search Trees.

1. ***Best Fit Bin Loading***

• New item is placed in a bin where it fits the tightest. If it does not fit in any bin, then start a new bin.

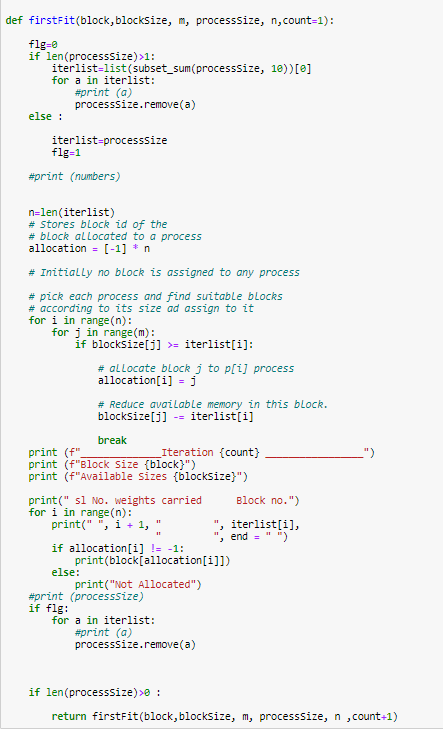
• Can be implemented in O(n log n) time, by using a balanced binary tree storing bins ordered by remaining capacity.

**When we compare First Fit Bin Loading algorithm and Best Fir Bin Loading algorithm, for the given use case we recommend using First Fit Bin Loading Algorithm, it used min number of rooms and maximum utilization in the selected rooms.**

**For selecting the packages for loading into robot and for tracking the available free space we are using single dimensional array.**

1. **Implement the algorithm in PYTHON**

*First Fit Algorithm:*



*Best Fit Algorithm:*



1. **Print the number of commutes by the robot, the weight carried by the robot in each commute, the number of rooms used to store the contingency and the remaining space for the store keeper’s reference.**

*Output for Best Fit Algorithm:*

**------------Best Fit --------------------**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 1** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [46, 35, 28, 25, 11, 4]

sl No. weights carried Block no.

1 9 20

2 1 5

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 2** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [46, 35, 28, 25, 1, 4]

sl No. weights carried Block no.

1 9 20

2 1 20

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 3** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [46, 35, 28, 16, 0, 4]

sl No. weights carried Block no.

1 9 25

2 1 20

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 4** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [46, 35, 28, 7, 0, 3]

sl No. weights carried Block no.

1 9 25

2 1 5

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 5** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [46, 35, 28, 0, 0, 0]

sl No. weights carried Block no.

1 7 25

2 2 5

3 1 5

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 6** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [46, 35, 18, 0, 0, 0]

sl No. weights carried Block no.

1 7 28

2 2 28

3 1 28

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 7** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [46, 35, 8, 0, 0, 0]

sl No. weights carried Block no.

1 6 28

2 4 28

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 8** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [46, 31, 2, 0, 0, 0]

sl No. weights carried Block no.

1 6 28

2 4 35

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 9** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [46, 21, 2, 0, 0, 0]

sl No. weights carried Block no.

1 6 35

2 4 35

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 10** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [46, 11, 2, 0, 0, 0]

sl No. weights carried Block no.

1 5 35

2 5 35

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 11** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [46, 3, 0, 0, 0, 0]

sl No. weights carried Block no.

1 4 35

2 4 35

3 2 28

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 12** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [38, 1, 0, 0, 0, 0]

sl No. weights carried Block no.

1 4 46

2 4 46

3 2 35

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 13** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [29, 0, 0, 0, 0, 0]

sl No. weights carried Block no.

1 2 46

2 1 35

3 1 46

4 1 46

5 1 46

6 1 46

7 1 46

8 1 46

9 1 46

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 14** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

**Available Sizes [28, 0, 0, 0, 0, 0]**

sl No. weights carried Block no.

1 1 46

*Output for First Fir Algorithm:*

**------------First Fit --------------------**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 1** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [36, 35, 28, 25, 20, 5]

sl No. weights carried Block no.

1 9 46

2 1 46

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 2** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [26, 35, 28, 25, 20, 5]

sl No. weights carried Block no.

1 9 46

2 1 46

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 3** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [16, 35, 28, 25, 20, 5]

sl No. weights carried Block no.

1 9 46

2 1 46

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 4** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [6, 35, 28, 25, 20, 5]

sl No. weights carried Block no.

1 9 46

2 1 46

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 5** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [3, 28, 28, 25, 20, 5]

sl No. weights carried Block no.

1 7 35

2 2 46

3 1 46

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 6** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [0, 21, 28, 25, 20, 5]

sl No. weights carried Block no.

1 7 35

2 2 46

3 1 46

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 7** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [0, 11, 28, 25, 20, 5]

sl No. weights carried Block no.

1 6 35

2 4 35

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 8** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [0, 1, 28, 25, 20, 5]

sl No. weights carried Block no.

1 6 35

2 4 35

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 9** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [0, 1, 18, 25, 20, 5]

sl No. weights carried Block no.

1 6 28

2 4 28

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 10** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [0, 1, 8, 25, 20, 5]

sl No. weights carried Block no.

1 5 28

2 5 28

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 11** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [0, 1, 0, 23, 20, 5]

sl No. weights carried Block no.

1 4 28

2 4 28

3 2 25

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 12** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [0, 1, 0, 13, 20, 5]

sl No. weights carried Block no.

1 4 25

2 4 25

3 2 25

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 13** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

Available Sizes [0, 0, 0, 4, 20, 5]

sl No. weights carried Block no.

1 2 25

2 1 35

3 1 25

4 1 25

5 1 25

6 1 25

7 1 25

8 1 25

9 1 25

\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Iteration 14** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Block Size [46, 35, 28, 25, 20, 5]

**Available Sizes [0, 0, 0, 3, 20, 5]**

sl No. weights carried Block no.

1 1 25