# 

# 

# **Design Document - Product Photoshoot**

# 

# Table of Contents

[**Table of Contents**](#_heading=h.qwbmaqlnjqxk) **1**

[Problem statement:](#_heading=h.qasoe6o7zl2e) 2

[Why Are Greedy Algorithms Called Greedy:](#_heading=h.ygfwhii1r0mk) 3

[Problem Assumptions:](#_heading=h.b4kce0p4dzfx) 3

[Solution Details:](#_heading=h.wp2vh84juxlg) 3

[Code Functions :](#_heading=h.32bti6oblaxk) 5

[Overall code flow](#_heading=h.erq7plfig1k3) 8

|  |
| --- |
| Problem statement: |

A famous product photographer Xavier is in high demand and is being approached by various companies to get their products photographed. Xavier does all of this product shoots at his studio and wants to make sure that he maximizes his returns by covering as many photoshoots in a day as possible. For a product shoot to be conducted, the product first has to be staged and then photographed. He has a team who stages the products and he does the photoshoots. Depending on the product, it takes different amounts of time to stage and varied time to complete the shoot. His team can stage another product in another part of the studio while Xavier is busy with a photoshoot of a previously staged product. Assuming that his team can work on only one product at a time before they move to staging another and Xavier also finishes one product photoshoot before moving to the next, you are expected to help Xavier make best use of his team and his time by deciding the order in which the products gets staged and photographed such that they finish the photoshoot of all products as quickly as possible.

Requirements:

1. Formulate and effective algorithm using Greedy Method to arrange the product staging and

photoshoot in such a way that total time taken for all products is minimized.

2. Analyse the time complexity of your algorithm.

|  |
| --- |
| Why Are Greedy Algorithms Called Greedy: |

Greedy algorithms are greedy. They do not look into the future to decide the global optimal solution.It looks for simple, easy-to-implement solutions to complex, multi-step problems by deciding which next step will provide the most obvious benefit. This means that the overall optimal solution may differ from the solution the algorithm chooses.

Greedy algorithms are often used in ad hoc mobile networking to efficiently route packets with the fewest number of hops and the shortest delay possible. They are also used in machine learning, business intelligence (BI), artificial intelligence (AI) and programming.

|  |
| --- |
| Problem Assumptions: |

1. Time is in minutes for staging and photo-shoot
2. Separate team for staging and Xavier does only photo-shoot. And both can work independently and parallely.
3. Team that does staging work on one product at a time. And move to another after completing one.
4. Photoshoot of product can only begin once staging task of that product is completed.

|  |
| --- |
| Solution Details: |

A **Simple Solution** would be to generate all combinations of given set of jobs and check total photoshoot and idle time of photographer.Choose combination for which the photoshoot and idle time is minimal. The time complexity of this solution is exponential. 

Per Greedy approach, we are targeting to start photoshoot time as soon as possible. This way initial idle time for Xavier will be minimal. To do that, we are ordering the product in ascending order for staging time i.e. small to maximum

**Sample input**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Product** | **A** | **B** | **C** | **D** | **E** | **F** |
| **Staging** | 20 | 30 | 45 | 60 | 20 | 10 |
| **Photo** | 30 | 30 | 15 | 20 | 40 | 60 |

# 

# **Staging Time Before sorting**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 20 [A] | 30 [B] | 45 [C] | 60 [D] | 20 [E] | 10 [F] |



# **Data sorted based on Staging Time**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Product** | **F** | **E** | **A** | **B** | **C** | **D** |
| **Staging** | 10 | 20 | 20 | 30 | 45 | 60 |
| **Photo** | 60 | 40 | 30 | 30 | 15 | 20 |

# 



# **Now Further while calculating the total Idle time and photoshoot time**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **10** | 70 | 110 | 140 | 170 | 185 | 205 |

# 10 > Total Idle Time

# 205 > Total Photoshoot Time

|  |
| --- |
| Code Functions : |

### sortJobScheduling(arr,short\_key)

In this functionality we are sorting input data based on the input column which is staging time. **The function sorts the products by increasing order of stage time.**

**This function makes use of merge sort which has lowest time complexity which is n\*log(n)**

# **Code Snippet -**

|  |
| --- |
| **# sorting input data based on the input column**  **def sortJobScheduling(arr, sort\_key=1):**  **final\_arr = []**  **arr1 = []**  **arr2 = []**  **if len(arr) < 2:**  **final\_arr = arr**  **else:**  **arr1 = merge\_sort(arr[0: round(len(arr) / 2)])**  **arr2 = merge\_sort(arr[round(len(arr) / 2):])**  **arr1\_inc = 0**  **arr2\_inc = 0**  **while 1 == 1:**  **if arr1[arr1\_inc][sort\_key] < arr2[arr2\_inc][sort\_key]:**  **final\_arr.append(arr1[arr1\_inc])**  **if arr1\_inc < len(arr1) - 1:**  **arr1\_inc += 1**  **else:**  **for i in arr2[arr2\_inc:]:**  **final\_arr.append(i)**  **break**  **else:**  **final\_arr.append(arr2[arr2\_inc])**  **if arr2\_inc < len(arr2) - 1:**  **arr2\_inc += 1**  **else:**  **for i in arr1[arr1\_inc:]:**  **final\_arr.append(i)**  **break**  **return final\_arr** |

* **calculateCompletionTime(input\_data):**

This function takes an array containing staging and photoshoot time for all the products. And it calculates photographer idle time and total photoshoot time.

**Time complexity is O(n)** as it's going to iterate through the entire dataset.

As we might need to traverse through the dataset and if possible recheck.

So, Time complexity would be O(nlogn), where n is the number of elements or column

# **Code Snippet -**

|  |
| --- |
| **# calculating photographer idle time and total elapsed time**  **def calculateCompletionTime(input\_data):**  **idle\_time, photoshoot\_time, stage\_time = 0, 0, 0**  **for i in input\_data:**  **stage\_time += i[1]**  **temp\_idle\_time = stage\_time - photoshoot\_time**  **if temp\_idle\_time > 0:**  **photoshoot\_time += i[2] + temp\_idle\_time**  **idle\_time += temp\_idle\_time**  **else:**  **photoshoot\_time += i[2]**  **return photoshoot\_time, idle\_time** |

* **outputWriter(output\_file\_path, arr, elapse\_time, idle\_time):**

This function writes the job sequence, photoshoot time and idle time as per the requirement in the output file.

**Time complexity for this function O(n) as it scans array to print array elements in formatted string**

# **Code Snippet -**

|  |
| --- |
| **# This function is use for to print the job sequence**  **def outputWriter(output\_file\_path, arr, elapse\_time, idle\_time):**  **out = open(output\_file\_path, 'w')**  **out.write("Product Sequence: " + ', '.join([str(x[0]) for x in arr]))**  **out.write("\n")**  **out.write("Total time to complete photoshoot: {0} minutes\n".format(elapse\_time))**  **out.write("Idle time for Xavier: {0} minutes".format(idle\_time))**  **print("Output successfully written in the output file located at: ", output\_file\_path)** |

|  |
| --- |
| Overall code flow |

1. Reads input file containing product detail from [inputPS7.txt](https://github.com/shivabansal1046/BITSAssignments/blob/main/DSAD_Group_102/Assignment_1_PS18_Grade_Book/final_delivery/inputPS18.txt)
2. Parse input file and creates an array containing products and their respective stage and photoshoot time.
3. Sort all jobs in increasing order of stage time.
4. Calculate total photoshoot time and idle time.
5. writes the output calculated in step 3 to output file [outputPS7.txt](https://github.com/shivabansal1046/BITSAssignments/blob/main/DSAD_Group_102/Assignment_1_PS18_Grade_Book/final_delivery/outputPS18.txt)