

[August 4, 2020](https://geekstocode.com/the-painters-partition-problem/)

**The Painter’s Partition Problem**

In this post, we are going to discuss a famous Binary Search problem that is **The Painter’s Partition Problem**. Also, I am going to discuss a general way to tackle some similar types of problems on Binary Search.  
  
So, let’s start..!!!

Table of Contents

* [Problem Statement](https://geekstocode.com/the-painters-partition-problem/#Problem_Statement)
* [Approach for the painter’s partition problem](https://geekstocode.com/the-painters-partition-problem/#Approach_for_the_painter%E2%80%99s_partition_problem)
  + [What is Search Space ?](https://geekstocode.com/the-painters-partition-problem/#What_is_Search_Space)
  + [Binary Search on Search space](https://geekstocode.com/the-painters-partition-problem/#Binary_Search_on_Search_space)
* [Code in C++ | The Painter’s Partition Problem](https://geekstocode.com/the-painters-partition-problem/#Code_in_C++_%7C_The_Painter%E2%80%99s_Partition_Problem)

**Problem Statement**

There are paint **n** boards of length **{l1, l2…ln}** and there are **k painters** available. Each painter takes **1**unit of time to paint **1** unit of the board.  
  
The problem is to find the **minimum time** to get this job done under the constraints that any painter will only paint continuous sections of boards.

**Example 1 :**

**Input** : k = 10, Board[] = [1, 8, 11, 3]

**Output** : 11

Since there are 10 painters and each painter take 1 sec time to paint 1 unit of board, we will give task of painting Board[] **= {1, 8}** to painter1,Board[]**= {11}** to painter2 and Board[] **= {3}** to painter3.  
  
If all painters will start there task at the same time the maximum of **11** unit time will be required to paint all boards.

**Example 2**

**Input** : k = 2, Board[] = [10, 20, 30, 40]

**Output** : 60

Here painter1 will paint Board[] = **{10, 20, 30}** and painter2 will paint Board[] = **{40}** and this is minimum way we can divide the task.

**Approach for the painter’s partition problem**

This problem can be solved by Binary Search and it is one of the problems of Binary Search patterns.  
  
Let’s discuss how we can apply a binary search on this problem.  
  
The main idea is to apply binary search on the **search space** and according to the problem we have to reduce the search space which will finally give us the final result.

**What is Search Space ?**

**Search space** is the maximum range where the answer contains. For example, in [10, 20, 30, 40] what is maximum and minimum time will be required to paint all the board.  
  
The maximum time will be (10+20+30+40) and the minimum will be **40**. Forget about the number of painters, just think of the best and worst situations.  
  
So, basically our search space will be **[40 – 100]**. Isn’t it ?

**Binary Search on Search space**

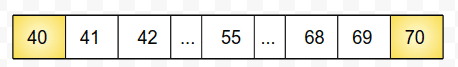
Let’s take an example,

**k = 2, Board[] = [10, 20, 30, 40]**

Search Space will be,

painter's partition problem

Now, let divide the search space, mid = 70 {(40+100)/2}.  
  
Now let’s assume that no painter will paint more than 70 units of the board.  
  
Board[] = [10, 20, 30, 40]  
  
1st painter will paint (10+20+30) unit board, 2nd painter will paint (40) unit board which concludes that a minimum of 2 painters will be required where each painter can paint at most 70 units of the board.  
  
Since K = 2, we can have a maximum of two painters. Now we know for 70 also two painters will be required.  
  
So, we can cancel the **second half** and now our search space will change to **[40, 70]** to check whether any less possible answer is there or not.  
  
**NOTE** our search space got reduced to half by binary search.



For, this search space we will see find mid = 55 and we will continue the same process.  
  
Again check for the case where no painter will paint more than **55 units** of the board and for that minimum 3 painters will be required.  
  
1st painter = [10, 20], 2nd painter = [30], 3rd painter = [40]  
  
Now, 55 will surely not be our answer so, we will neglect the left part, and surely answer lies on Search Space **[55, 70]**.  
  
We will continue this process by binary search and finally, we will get our answer which is**60** in this case.

**Code in C++ | The Painter’s Partition Problem**

Since we have discussed the idea, now let’s discuss the code.

int find(vector<int> &board, int at\_most)

{

int n = board.size();

int s = 0, painters = 1;

for (int i = 0; i < n; i++)

{

s += board[i];

if (s > at\_most)

{

s = board[i];

painters++;

}

}

return painters;

}

int partition(vector<int> &board, int k)

{

int n = board.size(), s = 0, m = 0;

for(int i = 0; i < n; i++)

{

m = max(m, board[i]);

s += board[i];

}

int low = m, high = s;

while (low < high)

{

int mid = low + (high - low) / 2;

int painters = find(board, mid);

if (painters <= k) high = mid;

else low = mid + 1;

}

return low;

}