

华为OD机试 – 处理器问题（Java & JS & Python）

原创

伏城之外

已于 2023-05-04 17:02:30 修改

13950

收藏 26

版权

分类专栏:

华为OD机试（Java & JS & Python）

华为OD机试2023A

文章标签:

算法

华为机试

JavaScript

Java

Python

OD

华为OD机试2... 同时被 2 个专栏收录

该专栏为热销专栏榜 第46名

¥49.90

¥99.90

253 订阅

131 篇文章

已订阅

题目描述

某公司研发了一款高性能AI处理器。每台物理设备具备8颗AI处理器，编号分别为0、1、2、3、4、5、6、7。

编号0-3的处理器处于同一个链路中，编号4-7的处理器处于另外一个链路中，不通链路中的处理器不能通信。

如下图所示。现给定服务器可用的处理器编号数组array，以及任务申请的处理器数量num，找出符合下列亲和性调度原则的芯片组合。

如果不存在符合要求的组合，则返回空列表。

亲和性调度原则：

- 如果申请处理器个数为1，则选择同一链路，剩余可用的处理器数量为1个的最佳，其次是剩余3个的为次佳，然后是剩余2个，最后是剩余4个。
- 如果申请处理器个数为2，则选择同一链路剩余可用的处理器数量2个的为最佳，其次是剩余4个，最后是剩余3个。
- 如果申请处理器个数为4，则必须选择同一链路剩余可用的处理器数量为4个。
- 如果申请处理器个数为8，则申请节点所有8个处理器。

提示：

1. 任务申请的处理器数量只能是1、2、4、8。
2. 编号0-3的处理器处于一个链路，编号4-7的处理器处于另外一个链路。
3. 处理器编号唯一，且不存在相同编号处理器。

输入描述

输入包含可用的处理器编号数组array，以及任务申请的处理器数量num两个部分。

第一行为array，第二行为num。例如：

[0, 1, 4, 5, 6, 7]

1

表示当前编号为0、1、4、5、6、7的处理器可用。任务申请1个处理器。

• 0 <= array.length <= 8

• 0 <= array[i] <= 7

• num in [1, 2, 4, 8]

输出描述

输出为组合列表，当array=[0, 1, 4, 5, 6, 7]，num=1时，输出为[[0],[1]]。

输入	[0, 1, 4, 5, 6, 7] 1
输出	[[0],[1]]
说明	根据第一条亲和性调度原则，在剩余两个处理器的链路（0, 1, 2, 3）中选择处理器。 由于只有0和1可用，则返回任意一颗处理器即可。

输入	[0, 1, 4, 5, 6, 7] 4
输出	[[4, 5, 6, 7]]
说明	根据第三条亲和性调度原则，必须选择同一链路剩余可用的处理器数量为4个的环

题目解析

用例中，链路link1=[0,1]，链路link2=[4,5,6,7]

现在要选1个处理器，则需要按照亲和性调度原则

如果申请处理器个数为1，则选择同一链路，剩余可用的处理器数量为1个的最佳，其次是剩余3个的为次佳，然后是剩余2个，最后是剩余4个。

最佳的是，找剩余可用1个处理器的链路，发现没有，link1剩余可用2，link2剩余可用4

其次的是，找剩余可用3个处理器的链路，发现没有

再次的是，找剩余可用2个处理器的链路，link1符合要求，即从0和1处理器中任选一个，有两种选择，可以使用dfs找对应组合。

关于 [回溯算法](#) 求解组合，可以看一下：

[LeetCode - 77 组合_算法的组合_伏城之外的博客-CSDN博客](#)

JavaScript算法源码

```
1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  const rl = readline.createInterface({
5    input: process.stdin,
6    output: process.stdout,
7  });
8
9  const lines = [];
10 rl.on("line", (line) => {
11   lines.push(line);
12
13   if (lines.length === 2) {
14     const arr = JSON.parse(lines[0]);
15     const num = lines[1];
16
17     console.log(getResult(arr, num));
18
19     lines.length = 0;
20   }
21 });
22
23 function getResult(arr, num) {
24   const link1 = [];
25   const link2 = [];
26
27   arr
28     .sort((a, b) => a - b)
29     .forEach((e) => {
30       e < 4 ? link1.push(e) : link2.push(e);
31     });
32
33   const ans = [];
34   const len1 = link1.length;
35   const len2 = link2.length;
36
37   switch (num) {
38     case "1":
39       if (len1 === 1 || len2 === 1) {
40         if (len1 === 1) dfs(link1, 0, 1, [], ans);
41         if (len2 === 1) dfs(link2, 0, 1, [], ans);
42       } else if (len1 === 3 || len2 === 3) {
43         if (len1 === 3) dfs(link1, 0, 1, [], ans);
44         if (len2 === 3) dfs(link2, 0, 1, [], ans);
45       } else if (len1 === 2 || len2 === 2) {
46         if (len1 === 2) dfs(link1, 0, 2, [], ans);
47         if (len2 === 2) dfs(link2, 0, 2, [], ans);
48       } else if (len1 === 4 || len2 === 4) {
49         if (len1 === 4) dfs(link1, 0, 1, [], ans);
50         if (len2 === 4) dfs(link2, 0, 1, [], ans);
51       }
52       break;
53     case "2":
54       if (len1 === 2 || len2 === 2) {
55         if (len1 === 2) dfs(link1, 0, 2, [], ans);
56         if (len2 === 2) dfs(link2, 0, 2, [], ans);
57       } else if (len1 === 4 || len2 === 4) {
58         if (len1 === 4) dfs(link1, 0, 2, [], ans);
59         if (len2 === 4) dfs(link2, 0, 2, [], ans);
60       } else if (len1 === 3 || len2 === 3) {
61         if (len1 === 3) dfs(link1, 0, 2, [], ans);
62         if (len2 === 3) dfs(link2, 0, 2, [], ans);
63       }
64       break;
65     case "4":
66       if (len1 === 4 || len2 === 4) {
67         if (len1 === 4) ans.push(link1);
68         if (len2 === 4) ans.push(link2);
69       }
70       break;
71     case "8":
72       if (len1 === 4 && len2 === 4) {
73         ans.push([...link1, ...link2]);
74       }
75       break;
76   }
77
78   return JSON.stringify(ans).split(",").join(" ");
79 }
80
81 function dfs(arr, index, level, path, res) {
82   if (path.length === level) {
83     return res.push([...path]);
84   }
85
86   for (let i = index; i < arr.length; i++) {
87     path.push(arr[i]);
88     dfs(arr, i + 1, level, path, res);
89     path.pop();
90   }
91 }
```

Java算法源码

```
1  import java.util.ArrayList;
2  import java.util.Arrays;
3  import java.util.Scanner;
4  import java.util.stream.Collectors;
5  import java.util.stream.Stream;
6
7  public class Main {
8    public static void main(String[] args) {
9      Scanner sc = new Scanner(System.in);
10
11      Integer[] arr =
12          Arrays.stream(sc.nextLine().split("[\\s\\S]"))
13              .filter(str -> !"".equals(str))
14              .map(Integer::parseInt)
15              .toArray(Integer[]::new);
16
17      String num = sc.next();
18
19      System.out.println(getResult(arr, num));
20    }
21
22    public static String getResult(Integer[] arr, String num) {
23      ArrayList<Integer> link1 = new ArrayList<>();
24      ArrayList<Integer> link2 = new ArrayList<>();
25
26      Arrays.sort(arr, (a, b) -> a - b);
27      for (Integer e : arr) {
28        if (e < 4) {
29          link1.add(e);
30        } else {
31          link2.add(e);
32        }
33      }
34
35      ArrayList<ArrayList<Integer>> ans = new ArrayList<>();
36      int len1 = link1.size();
37      int len2 = link2.size();
38
39      switch (num) {
40        case "1":
41          if (len1 == 1 || len2 == 1) {
42            if (len1 == 1) dfs(link1, 0, 1, new ArrayList<>(), ans);
43            if (len2 == 1) dfs(link2, 0, 1, new ArrayList<>(), ans);
44          } else if (len1 == 3 || len2 == 3) {
45            if (len1 == 3) dfs(link1, 0, 1, new ArrayList<>(), ans);
46            if (len2 == 3) dfs(link2, 0, 1, new ArrayList<>(), ans);
47          } else if (len1 == 2 || len2 == 2) {
48            if (len1 == 2) dfs(link1, 0, 2, new ArrayList<>(), ans);
49            if (len2 == 2) dfs(link2, 0, 2, new ArrayList<>(), ans);
50          } else if (len1 == 4 || len2 == 4) {
51            if (len1 == 4) dfs(link1, 0, 1, new ArrayList<>(), ans);
52            if (len2 == 4) dfs(link2, 0, 1, new ArrayList<>(), ans);
53          }
54          break;
55        case "2":
56          if (len1 == 2 || len2 == 2) {
57            if (len1 == 2) dfs(link1, 0, 2, new ArrayList<>(), ans);
58            if (len2 == 2) dfs(link2, 0, 2, new ArrayList<>(), ans);
59          } else if (len1 == 4 || len2 == 4) {
60            if (len1 == 4) dfs(link1, 0, 2, new ArrayList<>(), ans);
61            if (len2 == 4) dfs(link2, 0, 2, new ArrayList<>(), ans);
62          } else if (len1 == 3 || len2 == 3) {
63            if (len1 == 3) dfs(link1, 0, 2, new ArrayList<>(), ans);
64            if (len2 == 3) dfs(link2, 0, 2, new ArrayList<>(), ans);
65          }
66          break;
67        case "4":
68          if (len1 == 4 || len2 == 4) {
69            if (len1 == 4) ans.add(link1);
70            if (len2 == 4) ans.add(link2);
71          }
72          break;
73        case "8":
74          if (len1 == 4 && len2 == 4) {
75            ans.add(
76                Stream.concat(link1.stream(), link2.stream())
77                    .collect(Collectors.toCollection(ArrayList<Integer>::new)));
78          }
79          break;
80      }
81
82      return ans.toString();
83    }
84
85    public static void dfs(
86        ArrayList<Integer> arr,
87        int index,
88        int level,
89        ArrayList<Integer> path,
90        ArrayList<ArrayList<Integer>> res) {
91      if (path.size() == level) {
92        res.add(new ArrayList<>(path));
93        return;
94      }
95
96      for (int i = index; i < arr.size(); i++) {
97        path.add(arr.get(i));
98        dfs(arr, i + 1, level, path, res);
99        path.remove(path.size() - 1);
100      }
101    }
102 }
```

Python算法源码

```
1  # 输入获取
2  arr = eval(input())
3  num = int(input())
4
5
6  # 算法入口
7  def getResult(arr, num):
8      link1 = []
9      link2 = []
10
11      arr.sort()
12
13      for e in arr:
14         if e < 4:
15             link1.append(e)
16         else:
17             link2.append(e)
18
19      ans = []
20      len1 = len(link1)
21      len2 = len(link2)
22
23      if num == 1:
24         if len1 == 1 or len2 == 1:
25             if len1 == 1:
26                 dfs(link1, 0, 1, [], ans)
27             if len2 == 1:
28                 dfs(link2, 0, 1, [], ans)
29         elif len1 == 3 or len2 == 3:
30             if len1 == 3:
31                 dfs(link1, 0, 1, [], ans)
32             if len2 == 3:
33                 dfs(link2, 0, 1, [], ans)
34         elif len1 == 2 or len2 == 2:
35             if len1 == 2:
36                 dfs(link1, 0, 2, [], ans)
37             if len2 == 2:
38                 dfs(link2, 0, 2, [], ans)
39         elif len1 == 4 or len2 == 4:
40             if len1 == 4:
41                 dfs(link1, 0, 1, [], ans)
42             if len2 == 4:
43                 dfs(link2, 0, 1, [], ans)
44     elif num == 2:
45         if len1 == 2 or len2 == 2:
46             if len1 == 2:
47                 dfs(link1, 0, 2, [], ans)
48             if len2 == 2:
49                 dfs(link2, 0, 2, [], ans)
50         elif len1 == 4 or len2 == 4:
51             if len1 == 4:
52                 dfs(link1, 0, 2, [], ans)
53             if len2 == 4:
54                 dfs(link2, 0, 2, [], ans)
55         elif len1 == 3 or len2 == 3:
56             if len1 == 3:
57                 dfs(link1, 0, 2, [], ans)
58             if len2 == 3:
59                 dfs(link2, 0, 2, [], ans)
60     elif num == 4:
61         if len1 == 4 or len2 == 4:
62             if len1 == 4:
63                 ans.append(link1)
64             if len2 == 4:
65                 ans.append(link2)
66     elif num == 8:
67         if len1 == 4 and len2 == 4:
68             tmp = []
69             tmp.extend(link1)
70             tmp.extend(link2)
71             ans.append(tmp)
72
73     return ans
74
75
76 def dfs(arr, index, level, path, res):
77     if len(path) == level:
78         res.append(path[:])
79         return
80
81     for i in range(index, len(arr)):
82         path.append(arr[i])
83         dfs(arr, i + 1, level, path, res)
84         path.pop()
85
86
87 # 算法调用
88 print(getResult(arr, num))
```