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 ~~¥99.00~~ 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]]。

用例

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

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 2) {
    const arr = JSON.parse(lines[0]);
    const num = lines[1];

    console.log(getResult(arr, num));

    lines.length = 0;
  }
});

function getResult(arr, num) {
  const link1 = [];
  const link2 = [];

  arr
    .sort((a, b) => a - b)
    .forEach((e) => {
      e < 4 ? link1.push(e) : link2.push(e);
    });

  const ans = [];
  const len1 = link1.length;
  const len2 = link2.length;

  switch (num) {
    case "1":
      if (len1 === 1 || len2 === 1) {
        if (len1 === 1) dfs(link1, 0, 1, [], ans);
        if (len2 === 1) dfs(link2, 0, 1, [], ans);
      } else if (len1 === 3 || len2 === 3) {
        if (len1 === 3) dfs(link1, 0, 1, [], ans);
        if (len2 === 3) dfs(link2, 0, 1, [], ans);
      } else if (len1 === 2 || len2 === 2) {
        if (len1 === 2) dfs(link1, 0, 1, [], ans);
        if (len2 === 2) dfs(link2, 0, 1, [], ans);
      } else if (len1 === 4 || len2 === 4) {
        if (len1 === 4) dfs(link1, 0, 1, [], ans);
        if (len2 === 4) dfs(link2, 0, 1, [], ans);
      }
      break;
    case "2":
      if (len1 === 2 || len2 === 2) {
        if (len1 === 2) dfs(link1, 0, 2, [], ans);
        if (len2 === 2) dfs(link2, 0, 2, [], ans);
      } else if (len1 === 4 || len2 === 4) {
        if (len1 === 4) dfs(link1, 0, 2, [], ans);
        if (len2 === 4) dfs(link2, 0, 2, [], ans);
      } else if (len1 === 3 || len2 === 3) {
        if (len1 === 3) dfs(link1, 0, 2, [], ans);
        if (len2 === 3) dfs(link2, 0, 2, [], ans);
      }
      break;
    case "4":
      if (len1 === 4 || len2 === 4) {
        if (len1 === 4) ans.push(link1);
        if (len2 === 4) ans.push(link2);
      }
      break;
    case "8":
      if (len1 === 4 && len2 === 4) {
        ans.push([...link1, ...link2]);
      }
      break;
  }

  return JSON.stringify(ans).split(",").join(", ");
}

function dfs(arr, index, level, path, res) {
  if (path.length === level) {
    return res.push([...path]);
  }

  for (let i = index; i < arr.length; i++) {
    path.push(arr[i]);
    dfs(arr, i + 1, level, path, res);
    path.pop();
  }
}
```

Java算法源码

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    Integer[] arr =
        Arrays.stream(sc.nextLine().split("[\\[\\]\\,\\s]"))
            .filter(str -> !"".equals(str))
            .map(Integer::parseInt)
            .toArray(Integer[]::new);

    String num = sc.next();

    System.out.println(getResult(arr, num));
  }

  public static String getResult(Integer[] arr, String num) {
    ArrayList<Integer> link1 = new ArrayList<>();
    ArrayList<Integer> link2 = new ArrayList<>();

    Arrays.sort(arr, (a, b) -> a - b);
    for (Integer e : arr) {
      if (e < 4) {
        link1.add(e);
      } else {
        link2.add(e);
      }
    }

    ArrayList<ArrayList<Integer>> ans = new ArrayList<>();
    int len1 = link1.size();
    int len2 = link2.size();

    switch (num) {
      case "1":
        if (len1 == 1 || len2 == 1) {
          if (len1 == 1) dfs(link1, 0, 1, new ArrayList<>(), ans);
          if (len2 == 1) dfs(link2, 0, 1, new ArrayList<>(), ans);
        } else if (len1 == 3 || len2 == 3) {
          if (len1 == 3) dfs(link1, 0, 1, new ArrayList<>(), ans);
          if (len2 == 3) dfs(link2, 0, 1, new ArrayList<>(), ans);
        } else if (len1 == 2 || len2 == 2) {
          if (len1 == 2) dfs(link1, 0, 1, new ArrayList<>(), ans);
          if (len2 == 2) dfs(link2, 0, 1, new ArrayList<>(), ans);
        } else if (len1 == 4 || len2 == 4) {
          if (len1 == 4) dfs(link1, 0, 1, new ArrayList<>(), ans);
          if (len2 == 4) dfs(link2, 0, 1, new ArrayList<>(), ans);
        }
        break;
      case "2":
        if (len1 == 2 || len2 == 2) {
          if (len1 == 2) dfs(link1, 0, 2, new ArrayList<>(), ans);
          if (len2 == 2) dfs(link2, 0, 2, new ArrayList<>(), ans);
        } else if (len1 == 4 || len2 == 4) {
          if (len1 == 4) dfs(link1, 0, 2, new ArrayList<>(), ans);
          if (len2 == 4) dfs(link2, 0, 2, new ArrayList<>(), ans);
        } else if (len1 == 3 || len2 == 3) {
          if (len1 == 3) dfs(link1, 0, 2, new ArrayList<>(), ans);
          if (len2 == 3) dfs(link2, 0, 2, new ArrayList<>(), ans);
        }
        break;
      case "4":
        if (len1 == 4 || len2 == 4) {
          if (len1 == 4) ans.add(link1);
          if (len2 == 4) ans.add(link2);
        }
        break;
      case "8":
        if (len1 == 4 && len2 == 4) {
          ans.add(
              Stream.concat(link1.stream(), link2.stream())

               .collect(Collectors.toCollection(ArrayList<Integer>::new)));
        } 79 |       break;
    }

    return ans.toString();
  }

  public static void dfs(
      ArrayList<Integer> arr,
      int index,
      int level,
      ArrayList<Integer> path,
      ArrayList<ArrayList<Integer>> res) {
    if (path.size() == level) {
      res.add(new ArrayList<>(path));
      return;
    }

    for (int i = index; i < arr.size(); i++) {
      path.add(arr.get(i));
      dfs(arr, i + 1, level, path, res);
      path.remove(path.size() - 1);
    }
  }
}
```

Python算法源码

```python
# 输入获取
arr = eval(input())
num = int(input())


# 算法入口
def getResult(arr, num):
    link1 = []
    link2 = []

    arr.sort()

    for e in arr:
        if e < 4:
            link1.append(e)
        else:
            link2.append(e)

    ans = []
    len1 = len(link1)
    len2 = len(link2)

    if num == 1:
        if len1 == 1 or len2 == 1:
            if len1 == 1:
                dfs(link1, 0, 1, [], ans)
            if len2 == 1:
                dfs(link2, 0, 1, [], ans)
        elif len1 == 3 or len2 == 3:
            if len1 == 3:
                dfs(link1, 0, 1, [], ans)
            if len2 == 3:
                dfs(link2, 0, 1, [], ans)
        elif len1 == 2 or len2 == 2:
            if len1 == 2:
                dfs(link1, 0, 1, [], ans)
            if len2 == 2:
                dfs(link2, 0, 1, [], ans)
        elif len1 == 4 or len2 == 4:
            if len1 == 4:
                dfs(link1, 0, 1, [], ans)
            if len2 == 4:
                dfs(link2, 0, 1, [], ans)
    elif num == 2:
        if len1 == 2 or len2 == 2:
            if len1 == 2:
                dfs(link1, 0, 2, [], ans)
            if len2 == 2:
                dfs(link2, 0, 2, [], ans)
        elif len1 == 4 or len2 == 4:
            if len1 == 4:
                dfs(link1, 0, 2, [], ans)
            if len2 == 4:
                dfs(link2, 0, 2, [], ans)
        elif len1 == 3 or len2 == 3:
            if len1 == 3:
                dfs(link1, 0, 2, [], ans)
            if len2 == 3:
                dfs(link2, 0, 2, [], ans)
    elif num == 4:
        if len1 == 4 or len2 == 4:
            if len1 == 4:
                ans.append(link1)
            if len2 == 4:
                ans.append(link2)
    elif num == 8:
        if len1 == 4 and len2 == 4:
            tmp = []
            tmp.extend(link1)
            tmp.extend(link2)
            ans.append(tmp)

    return ans


def dfs(arr, index, level, path, res):
    if len(path) == level:
        res.append(path[:])
        return

    for i in range(index, len(arr)):
        path.append(arr[i])
        dfs(arr, i + 1, level, path, res)
        path.pop()


# 算法调用
print(getResult(arr, num))
```