五大算法之分支定界法 2——通过剪枝提升效率

作者：yanghang
原文链接：https://ld246.com/article/1482490664386
来源网站：链滴

上篇博客（五大算法之分支定界法）介绍了使用分支定界解决装载问题，该算法的时间、空间复杂度均为2^n，这篇博客考虑如何改进该算法，提升效率。
每次加入活节点之前，都需要考虑一下当前装载重量加上余货物总重量是否大于当前最优解，如果不是的话则表明该分支不可能在产生最优解，可以去除。也根据这个约束条件进行剪枝。
改进后运行w={ 10, 2, 5, 7, 5, 9, 4, 3, 2, 5, 3, 3, 4, 2, 3, 4, , 3, 4, 1, 2, 1, 2, 1, 2, 1 }实例，需要1838ms，没有优化之前的算法需要27776ms，效率提升明显。
代码：
<pre class="brush: java">package test;

import java.util.LinkedList;

/**

- 用分支定界法求装载问题，改进版
-
- @author yanghang
-

*/
public class Fenzhidingjie2 {

private static float[] w = { 10, 2, 5, 7, 5, 9, 4, 3, 2, 5, 3, 3, 4, 2, 3,
        4, 3, 3, 4, 1, 2, 1, 2, 1, 2, 1 }; // 货箱质量数组
private static int n = w.length; // 货箱数量
private static float c1 = 50; // 第一艘船的载重量
private static float c2 = 50; // 第二艘船的载重量

private static float bestw; // 第一艘船的最大装载
private static float ew; // 当前船的装载量
private static LinkedList<Float> mq = new LinkedList<Float>(); // FIFO队列

/**
 * 最优装载值
 *
 * @param c
 */
public static float maxLoading(float c) {
    mq.addLast(new Float(-1)); // 初始化结点队列，标记分层
    int i = 0; // E-结点的层
    ew = 0; // 当前船的装载量
    bestw = 0; // 目前的最优值
    // 剩余货物的重量，用于剪枝
    float r = 0;
    for (float f : w) {
        r += f;
    }
    r -= w[0];

```
    while (!mq.isEmpty()) { // 搜索子集空间树
        // 检查E-结点的左孩子，货箱i是否可以装载
        if (ew + w[i] &lt;= c) {
            if (ew + w[i] &gt; bestw) {
                bestw = ew + w[i]; // 更新最优值
            }
            addLiveNode(ew + w[i], i); // 货箱i可以装载
        }
        // 剪枝，当前重量加剩余重量大于当前最优值才继续，否则该分支不可能产生最优解
        if (ew + r &gt; bestw) {
            addLiveNode(ew, i); // 右孩子总是可行的，不装载货物i
        }
        ew = (Float) mq.removeFirst(); // 取下一个结点

        if (ew == -1) { // 到达层的尾部
            if (mq.isEmpty()) {
                return bestw;
            }
            mq.addLast(new Float(-1));
            ew = (Float) mq.removeFirst(); // 取下一个结点
            i++; // ew的层
            r -= w[i]; // 更新剩余重量
        }
    }
    return bestw;
}

/**
 * 入队
 *
 * @param wt
 * @param i
 */
public static void addLiveNode(float wt, int i) {
    if (i == n - 1) { // 下标从0开始，是叶子
        if (wt &gt; bestw) {
            bestw = wt;
        }
    } else { // 不是叶子
        mq.addLast(new Float(wt));
    }
}

public static void main(String[] args) {
    // TODO Auto-generated method stub
    // 所有货箱的重量之和
    long start = System.currentTimeMillis();

    float s = 0;
    for (float f : w) {
        s += f;
    }
    if (s &lt;= c1 || s &lt;= c2) {
        System.out.println("need only one ship!");
```

```
    }
    if (s &gt; c1 + c2) {
        System.out.println("no solution!");
        return;
    }

    float bestw = Fenzhidingjie2.maxLoading(c1);

    if (s - bestw &lt;= c2) {
        System.out.println("The first ship loading " + bestw);
        System.out.println("The second ship loading " + (s - bestw));
    } else {
        System.out.println("no solution!");
    }
    long end = System.currentTimeMillis();
    System.out.println(end - start);
}

}
</pre>
```