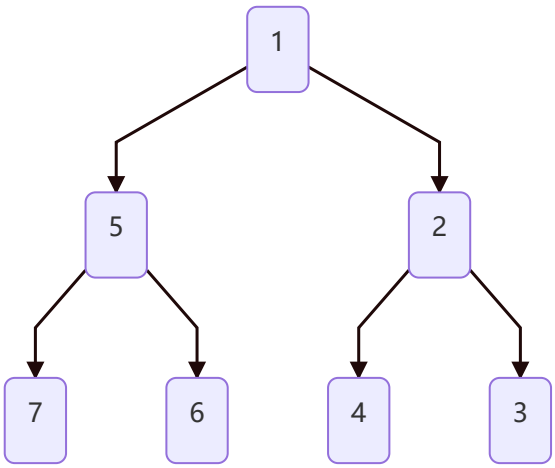
6-1：

> *栈式分支限界法将活结点表以后进先出（LIFO）的方式存储于栈中。试设计一个解0-1背包问题的栈式分支限界法，并说明栈式分支限界法与回溯法的区别。*

题目分析

栈式分支限界法的特点是后进先出，也就是如下图所示进行子集树的遍历：



可以发现，遍历的顺序与回溯法完全一致，可以借用回溯法的 dfs，使用栈来完成本题。

在本题中，使用上界函数进行优化，即：

$$currentprofit + maxprofit \leq bestprofit \tag{1}$$

其中，$currentprofit$ 是当前这个结点处已经装入的价值，$maxprofit$ 是剩余未选取物品的最大重量，$bestprofit$ 是最优解。

分支限界法与回溯法的区别：栈式分支限界法在搜索子集树的顺序与回溯法并无二致（如上图，若使用回溯法，且每次优先深搜右子树），但二者在扩展结点及活结点表的更新上略有差异：

- 回溯法每次只往活结点表里加入一个结点，即每次都只在递归调用dfs时加入下一个结点，并在搜索到叶子结点 将叶结点抛弃，之后返回至上一层，将刚刚已经做过扩展结点的结点再次当成扩展结点，遍历下一个叶子结点。
- 分支限界法每次加入多个活结点，例如上图中遍历结点1后，将2与5两个结点同时加入栈中，此时两个结点均已进入活结点表。而在搜索到叶子结点时，不必返回上一层，而是直接在栈内取下一个元素即可，也就是说，分支限界法中的每个结点均只有一次作为扩展结点的机会。

代码实现

详见 6-1 Code.cpp

输出示例

输入格式如下：

- 先输入一行两个整数 n, C，n 为物品的数量，C 为背包的最大承重
- 再输入两行，第一行输入 n 个物品的重量 $w[i]$，第二行输入 n 个物品的价值 $v[i]$

```
输入1：
7 25
4 6 8 3 6 7 4
9 3 7 9 8 4 7
输出：
背包的最大价值为：40
选取的物品编号为：1 3 4 5 7

输入2：
3 30
30 18  12
2  1   3
输出：
背包的最大价值为：4
选取的物品编号为：2 3

输入3：
5 1
3 6 2 4 8
3 4 7 1 2
输出：
背包的最大价值为：0
背包无法承载任何一件物品。
```

算法分析

- 输入部分的时间复杂度为 $O(n)$。
- 更新上界函数的时间复杂度为 $O(1)$，因此栈式分支限界法的时间复杂度为 $O(2^n)$，即子集树的活结点数。

因此，本算法的时间复杂度为 $O(2^n)$，可以看到与回溯法的 $O(n2^n)$差了一个 n，其原因在于该算法动态更新上界函数，即将上界存在了每个结点里面，用空间换时间，所以空间复杂度会比回溯法高。