**D. 市集採買**

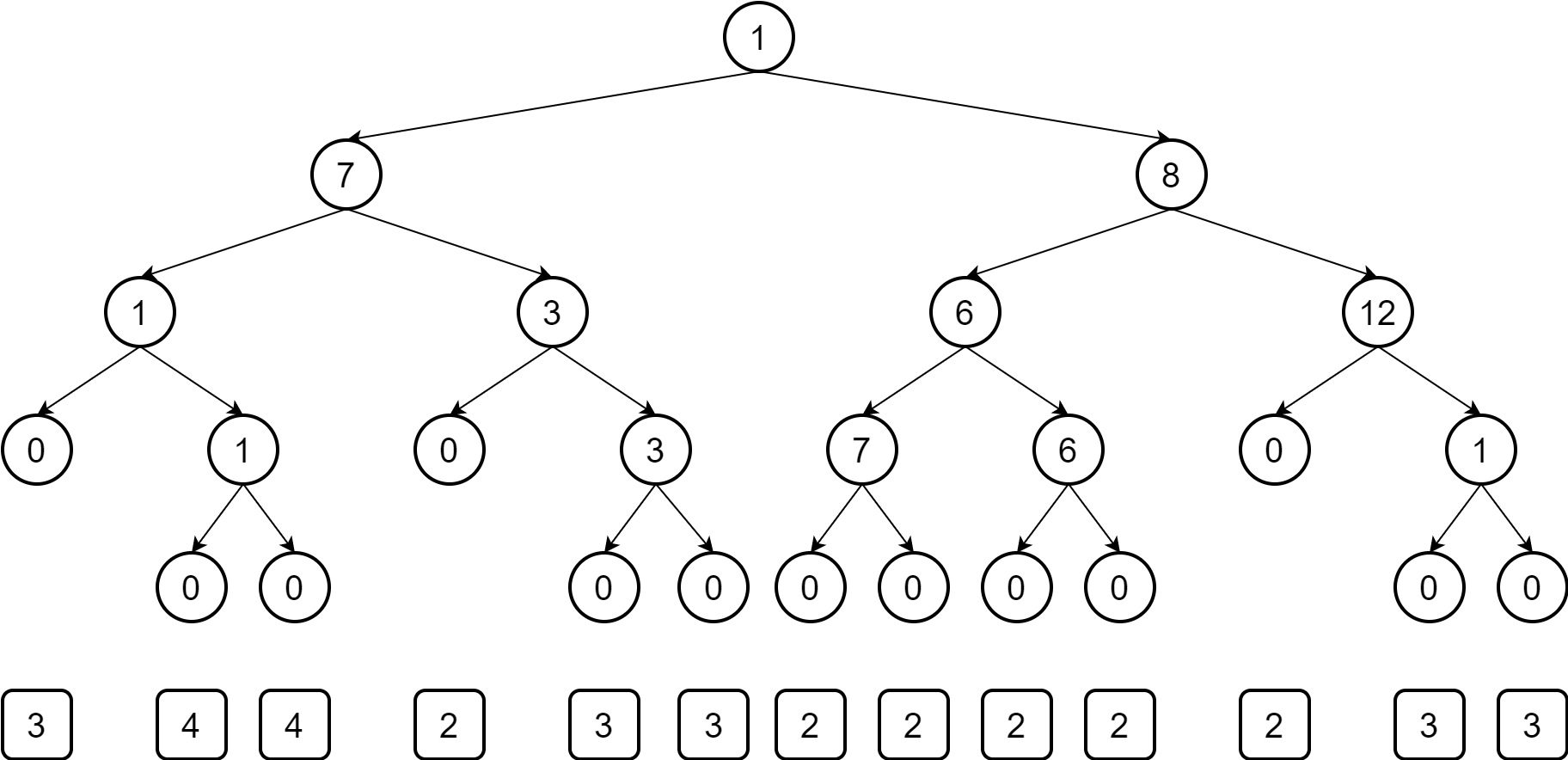
　　終端雞市集，是一個以二元樹為原型，打造出的市集，強大的設計理念，吸引許多攤販進駐。所有攤販都被安排在除了葉節點以外的節點，而葉節點只會有「已經到盡頭」的告示牌。攤販們被規定只能在攤位上賣一件物品，並且需要標明這件物品的大小，以便大家放心採買。

烏骨雞和放山雞放學後，決定去終端雞市集採買，烏骨雞想要盡可能在不超過背包容量下，買到最多物品，但放山雞太懶惰，不想逛完整個市集，最後，兩雞達成協議，只會逛市集的其中一條路線，然後原路折返。

　　即便如此，烏骨雞還是想盡可能買到最多物品，因此牠想設計一個程式，輸入背包容量後，再以前序輸入市集攤販所賣物品之大小，就能獲得一條能買最多物品的路線。但因為可能同時存在多條路線，烏骨雞又有選擇困難，因此輸出永遠只會輸出最右邊那條。

　　以終端雞市集為例，假設烏骨雞背包容量為10，以前序輸入市集攤販所賣物品之大小為「1 7 1 0 1 0 0 3 0 3 0 0 8 6 7 0 0 6 0 0 12 0 1 0 0」，其中0為「已經到盡頭」的告示牌。而樹的圖如下圖所示，圓形內數字代表攤販所賣物品之大小，最底下的矩形內數字代表該條路在不超過背包容量下，所能買到最多物品數量，而輸出要選擇最多物品又最右邊的路線，所以輸出為「1 7 1 1 0」。

　　由於烏骨雞能力不足，因此請你依所給條件，幫烏骨雞寫一個程式來達成目的。

****

**輸入格式**

　　第一行有一個不超過2147483647的正整數，代表烏骨雞的背包容量，第二行有 2n個攤販所賣物品之大小，每個攤販所賣物品之大小為一個不超過100000的正整數。保證n不超過20且二元樹的深度不超過20層。

**輸出格式**

　　輸出只有一行，為一條能買最多物品且最右邊的路線。

**測試資料**

|  |  |
| --- | --- |
| 輸入範例1  7  3 4 0 0 5 0 0 | 輸出範例1  3 4 0 |
| 輸入範例2  10  1 7 1 0 1 0 0 3 0 3 0 0 8 6 7 0 0 6 0 0 12 0 1 0 0 | 輸出範例2  1 7 1 1 0 |

**測試資料說明**

　　範例測資1中，總共有四條路徑，由左到右分別為「3 4 0」「3 4 0」「3 5 0」「3 5 0」，前兩條路徑能買到最多物品數量為2，後兩條為1，又選擇最右邊的路徑，故輸出「3 4 0」。

　　範例測資2請見題目敘述以及圖片。

**配分**

|  |  |  |  |
| --- | --- | --- | --- |
| **記憶體限制** | 64MBytes | **評分方式** | Tolerant (寬鬆比對) |
| **編號** | **配分** | **時間限制** | **敘述** |
| #0~#10 | 11% | 0.1s | 二元樹深度不超過5層 |
| #11~#29 | 19% | 0.1s | 二元樹深度不超過10層 |
| #30~#52 | 23% | 0.1s | 二元樹深度不超過15層 |
| #53~#99 | 47% | 0.1s | 無特別限制 |