

Stacking Boxes

99753506 林明慶
98703054 林曉智
98703018 莊士鋒

Agenda

- 103 – Stacking Boxes Problem
- Methods
- Analysis



103 – Stacking Boxes Problem(1/4)

- 在數學和計算機學中，一維或二維的概念是比較簡單的。
- 當維數不確定時，則就會變得比較複雜。
- 比方說，求高階微分方程和分析n維的超立方體拓撲結構中，高階微分方程要比維數為一階要複雜得多，而n維的超立方體拓撲結構則與較低階的同類具有非常大之相似之處。

103 – Stacking Boxes Problem(2/4)

- 給定一個n維“盒子”的維數。
- 在二維之情況下，box(2,3)可以表示一個長2個單位，寬3個單位的盒子。
- 在三維情況下，box(4,8,9)可以表示一個 $4 \times 8 \times 9$ （長寬高）的盒子。
- 在六維空間中，可能無法搞清楚box(4,5,6,7,8,9)是甚麼樣子，但我們可以分析盒子的性質，例如：求其維度之和。



103 – Stacking Boxes Problem(3/4)

- 要確定出盒子的最長嵌套串。也就是說一系列的盒子 b_1, b_2, \dots, b_k ，一個套一個，使所有的 b_i 都嵌套在 b_{i+1} 內。
- 對於一個盒子 $D=(d_1, d_2, \dots, d_n)$ 和另一個盒子 $E=(e_1, e_2, \dots, e_n)$ ，如果存在一種 d_i 的排列，使重排的每個維度的值都小於盒子E中對應維度的值，則盒子D可嵌入盒子E。這個過程和旋轉盒子D，看它是否能套入E的過程類似。
- 然而，若需滿足所有維度的排列，盒子D必需是可以扭曲的，而不僅僅是旋轉。

103 – Stacking Boxes Problem(4/4)

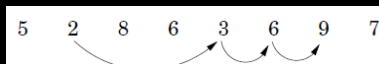
- 例如：
 1. 盒子 $D=(2,6)$ 可嵌入盒子 $E=(7,3)$ 是由於盒子可以排列為 $(6,2)$ ，使每個維度都小於E中的相應維度。
 2. 而盒子 $D=(9,5,7,3)$ 就不能嵌入盒子 $E=(2,10,6,8)$ ，因為不存在D的一種排列可以滿足嵌入的性質，但是 $F=(9,5,7,1)$ 就可以通過排列為 $(1,9,5,7)$ 而嵌入E。
- 我們正式的定義嵌套的關係如下：
盒子 $D=(d_1, d_2, \dots, d_n)$ 和 $E=(e_1, e_2, \dots, e_n)$ ，當存在一種 $1..n$ 的排列 p 使得 $(d_{p(1)}, d_{p(2)}, \dots, d_{p(n)})$ 能夠匹配 (e_1, e_2, \dots, e_n) ，即對於所有的 $1 \leq i \leq n$ 都有 $d_{p(i)} < e_i$ ，那麼就認為D可以嵌入E。

Methods 1: DAG (最長路徑問題)

- 先將盒子D=(d1, d2, ..., dn)和E=(e1, e2, ..., en)分別依小排到大排序好。
- 當存在一種1...n的排列p使得(d_{p(1)}, d_{p(2)}, ..., d_{p(n)})能夠匹配於盒子E=(e₁, e₂, ..., e_n)。即對於所有的1 ≤ i ≤ n都有d_{p(i)} < e_i，那麼就認為D可以嵌入E。
- 故我們假設若a”可嵌套於”b，那麼就是G[a][b]=1。然後這個問題就可以轉化成求一個最長路徑長度的大小。

Methods 2: LIS (最長遞增子序列)

- 給定一個序列a₁, a₂, ..., a_n
- 定義原序列中的子序列為原序列元素的一個子集，且子序列各元素的相對應順序不變a_{i₁}, a_{i₂}, ..., a_{i_n}，且1 ≤ i₁ < i₂ < ... < i_n ≤ n。由此找出給定序列的最長遞增子序列。
- 例：序列5,2,8,6,3,6,9,7的最長遞增子序列是2,3,6,9



Sample for factory

- 工廠貨品範例
- 家中空箱
- 孩子學習
- 物流士

Analysis

- DAG → O(n²)

```

//判斷a是否包含b
inline bool isBigger(int a, int b){
    for(int i=0; i<=a; ++i)
        if(a[i]==b[i])return false;
    return true;
}

//判斷a是否包含b
inline bool isBigger(int a, int b){
    for(int i=0; i<=a; ++i)
        if(a[i]==b[i])return false;
    return true;
}

int main(){
    int n;
    for(int i=0; i<=n; ++i)
        if(isBigger(i, i-1))
            d[i]=1;
}

int main(){
    int n;
    for(int i=0; i<=n; ++i)
        if(isBigger(i, i-1))
            d[i]=1;
}

int main(){
    int n;
    for(int i=0; i<=n; ++i)
        if(isBigger(i, i-1))
            d[i]=1;
}
    
```

Analysis

- LIS → O(n²)

```

struct Node{
    int no;
    int A[12];
    int s;
    void sort();
};

friend bool operator < (const Node& a, const Node& b){
    for(int i=0; i<a.s; ++i)
        if(a.A[i]>b.A[i])return false;
    return true;
}

Node a[12];

//判斷a是否包含b
inline bool isBigger(int a, int b){
    for(int i=0; i<=a; ++i)
        if(a[i]==b[i])return false;
    return true;
}

int main(){
    int n;
    for(int i=0; i<=n; ++i)
        if(isBigger(i, i-1))
            d[i]=1;
}

int mainVal=1, pos=0;
s[0]=0; d[0]=1;
for(int i=0; i<=n; ++i)
    d[i]=1; s[i]=1;
    for(int j=0; j<i; ++j)if(a[j][1]>a[i][1])
        d[i]=d[j]+1;
    if(d[i]>maxVal){
        pos=i;
        maxVal=d[i];
    }
}
    
```

分工方式

姓名	內容
林明慶	Video, Coding, PowerPoint, Analysis
林擘智	Collecting data of DAG
莊士鋒	Collecting data of LIS

~Thanks for your Attention~