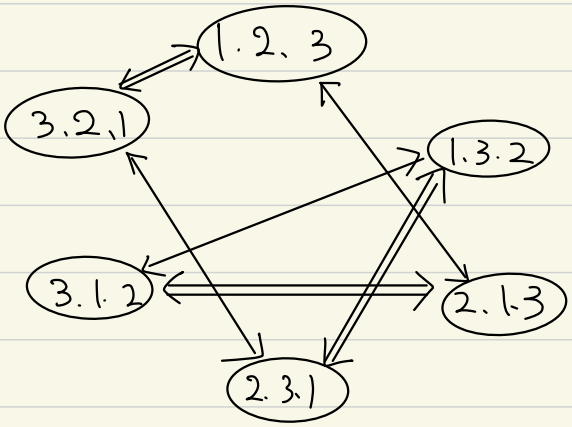
H27 [1]

(1) 考えうる頂点の個数は以下の $3! = 6$ 通り である.

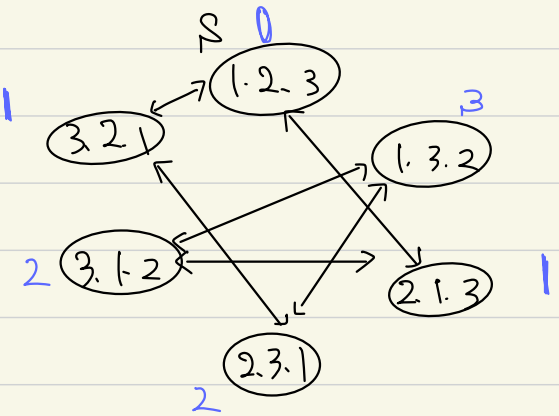
{1.2.3}, {1,3,2} {2,1,3}, {2.3.1}
{3.1.2}, {3.2.1}

またこれらの状態を頂点と見なした時の状態変化を有向辺にもつ状態遷移グラフを考えると以下の図のようになる。



(但し、— 線は $k=2$ の場合の遷移を, = 線は $k=3$ の場合の遷移を示すものとする)

(2) (1)で与えたグラフにおいて、辺をすべて逆向きに張ったグラフ G^{-1} を考える.
この G^{-1} において、$S' = (1.2.3)$ を初点とする最短路問題を考えることで、
もとのグラフにおいて、各状態から終点 (1.2.3) にたどり着くまでの最小手数を考える.



よってグラフより、{1.3.2} が最も手数を要し、その最小手数は 3 である.

(3) 考えうる頂点の個数は 4! = 24通りであり全て例示すると以下のようになる.

{1,2,3,4} {1,2,4,3} {1,3,2,4} {1,3,4,2} {1,4,2,3} {1,4,3,2}
{2,1,3,4} {2,1,4,3} {2,3,1,4} {2,3,4,1} {2,4,1,3} {2,4,3,1}
{3,1,2,4} {3,1,4,2} {3,2,1,4} {3,2,4,1} {3,4,1,2} {3,4,2,1}
{4,1,2,3} {4,1,3,2} {4,2,1,3} {4,2,3,1} {4,3,1,2} {4,3,2,1}

S

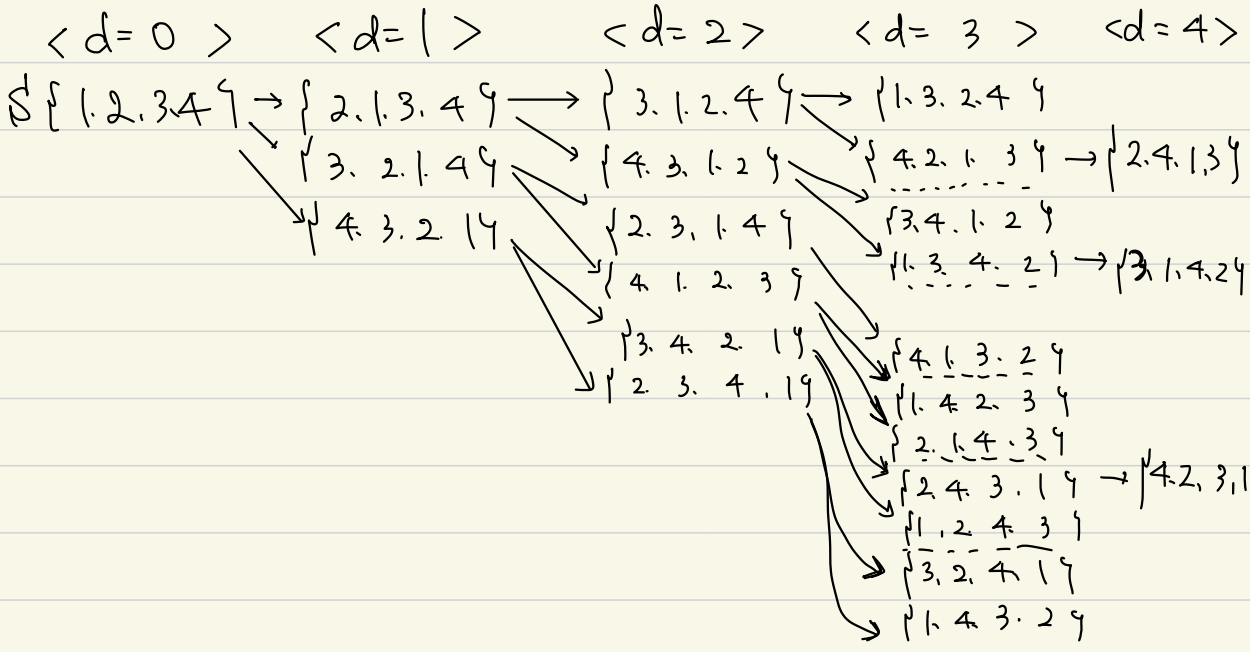
(2)と同様の考え方を用いる. 操作を逆から考えて, 終了状態{1,2,3,4}から何回か操作を行うことで, 各状態に遷移する上での最小コストを求めればよい.
ある状態から次の状態遷移は $k=2,3,4$ の3通りありうる.
1度Sから遷移が確認された頂点において, 次の遷移を考える必要はないので,
(∵過去にそこからの遷移を考慮済み)

S{1,2,3,4}から状態遷移グラフ上でBFS(幅優先探索)を行い、
各状態へ遷移する上での最小手数を確定させてやることを考える.

この結果をまとめると以下のようになる.

(d = 遷移に要する最小手数)

<d=0> <d=1> <d=2> <d=3> <d=4>



従って、{2,4,1,3}{3,1,4,2}{4,2,3,1}が最も手数を要し、その最小手数は4である。

(4) 時間計算量、空間計算量それぞれ$O(N^2)$, $O(N)$のアルゴリズムを示す。
また、必要な操作回数が高々2Nでおさえられることも示す。

これは、N枚に関する再帰的なアルゴリズムである。
その内容は、N枚のパンケーキを整列する上で、1番下に来るパンケーキから整列するように操作を行うというアイデアによるものである。

N枚のパンケーキが今ある状態遷移にあるとする。そして、

order[i] = 大きさiのパンケーキが上から何番目に位置するか

となる長さNの配列を定義し、初期状態に応じて初期化する。
これはO(N)で可能であることは明らかである。

さて、初めに述べたアイデアに沿ってパンケーキを並び替えるアルゴリズムの疑似コードを以下に示す。
なお、コード中の//(コメントアウト)にその内容の補足を示す

```
<Function>  sort(n)   // パンケーキの上n枚をソートする関数
    if  n=1  ⟶  return   //ソート済
    if  order[n] = n  ⟶  return sort(n-1)
                          //上n-1枚をソートするのでよい

    pos = order[n]
    ①(上からpos枚のパンケーキを手順でひっくり返す, つまりk=posとしてソートする.)
    例(上の更新にともない, 上からpos枚のパンケーキについてorderの要素を更新する.)
```

②（上から n枚のパンケーキを操作でひっくり返す、つまり k=n としてソートする）
(**)（上の更新にともない、上から n枚のパンケーキについて、orderの要素を更新する）

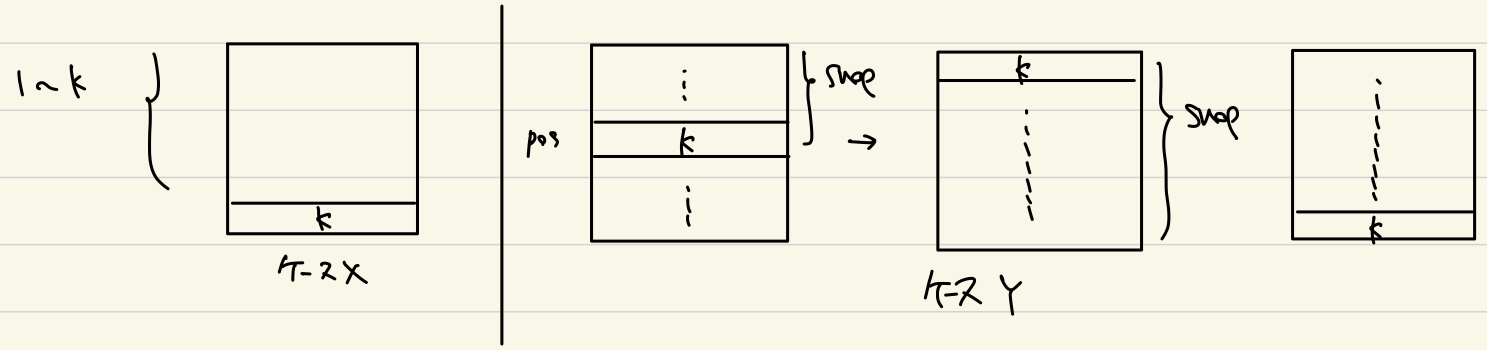
```
    // この時点で order[n] = n が確定する
    return sort(n-1)      // 残る上の n-1 枚をソートする
```

このアルゴリズムにおいて不変的に成立することは、

- sort(i) が呼ばれた時、上から i+1 番目以降の要素は整列済み

ということである。これが正しい場合、sort(1)が呼ばれた時点でパンケーキが整列済みであることは明らかである。

上を i に関する帰納法で示す。$i=n$ の時は自明。
$i=k+1$ で正しいとして、$i=k$ の場合を考える。



この場合、上k枚のパンケーキは {1,… k} の並び替えである。

ケースXのような場合成立は明らか
ケースYのような場合、2回の操作及び2回のorder更新後 条件を満たす（疑似コードの①② *, **）

以上より帰納的に示された。

最後に時間計算量を考える。sort(n)を呼んだ後、高々2回の操作と更新コスト O(N) の order 更新を行っているので、以下が成立する

$$O(sort(n)) = O(sort(n-1)) + O(n) \quad \cdots (\alpha)$$

(※) より

$O(\mathrm{sort}(n)) = O(n^2)$ が明らかに従う.

また, 高々 $2 \times N = 2N$ 回の操作でもとの状態から整列状態に直せることも明らか ■