Chương 6: Kỹ thuật Giảm để trị (Decrease-and-Conquer)

Giới thiệu chung

Giảm với lượng không đổi (*decrease by a constant*)

Kích thước dữ liệu giảm một lượng không đổi sau mỗi bước lặp, thường là 1. Thỉnh thoảng, con số này là 2.

Ví dụ: Tính a^n

$$f(n) = \begin{cases} f(n-1) \times a & nếu\ n > 1 \\ a & nếu\ n = 1 \end{cases}$$

Giảm với tỉ lệ không đổi (*decrease by a constant factor*)

Kích thước dữ liệu giảm bởi tỉ lệ α (thường là $\frac{1}{2}$) không đổi sau mỗi bước lặp.

Ví dụ: Tìm kiếm nhị phân. Lời giải với dữ liệu kích thước *n* cũng vẫn là lời giải với dữ liệu kích thước *n* / 2.

Ví dụ: Tính a^n

$$a^n = \begin{cases} (a^{n/2})^2 & Nếu\ n\ chẵn \\ \left(a^{\lfloor n/2 \rfloor}\right)^2 \times a & Nếu\ n\ lẻ\ và\ n \geq 1 \\ 1 & Nếu\ n = 0 \end{cases}$$

Chú ý: Công thức tính theo chia để trị là:

$$a^n = \begin{cases} a^{\lfloor n/2 \rfloor} \times a^{\lceil n/2 \rceil} & nếu\ n \geq 1 \\ 1 & nếu\ n = 0 \end{cases}$$

Ví dụ: Nhân $n \times m$

$$n \times m = \begin{cases} \frac{n}{2} \times 2m & nếu\ n\ chẵn \\ \left\lfloor \frac{n}{2} \right\rfloor \times 2m + m & nếu\ n\ lẻ \end{cases}$$

Chúng ta có thể cài đặt đệ qui hoặc vòng lặp với điểm dừng là: $1 \times m = m$.

Giảm với lượng biến đổi (*variable size decrease*)

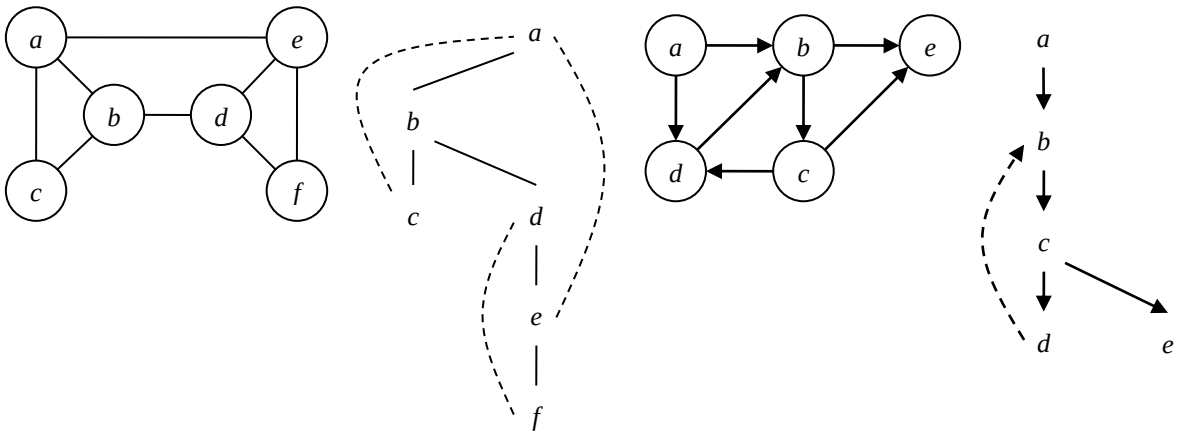
Kích thước của thể hiện của bài toán giảm với lượng không xác định sau mỗi lượt lặp của giải thuật.

Ví dụ: Giải thuật Euclid để tìm USCLN của hai số *a* và *b*.

$$\gcd(a, b) = \gcd(b, a \bmod b)$$

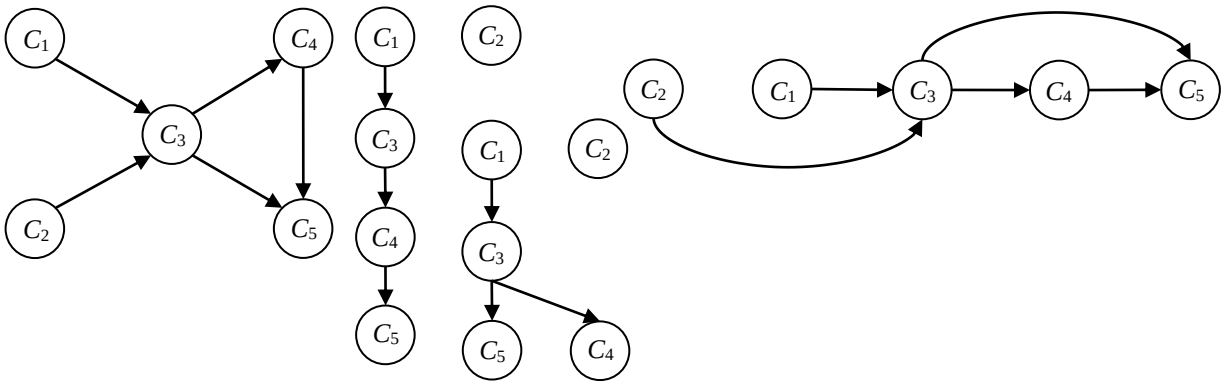
Sắp xếp Topology

***Một số khái niệm liên quan*:** chu trình, cạnh lùi, DAG

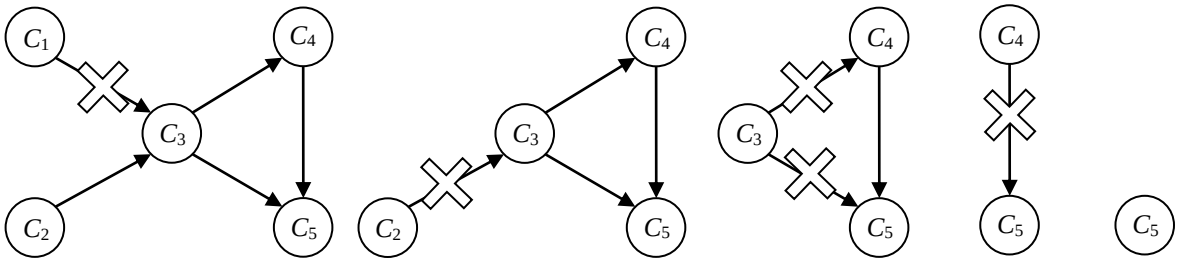


Mô tả vấn đề

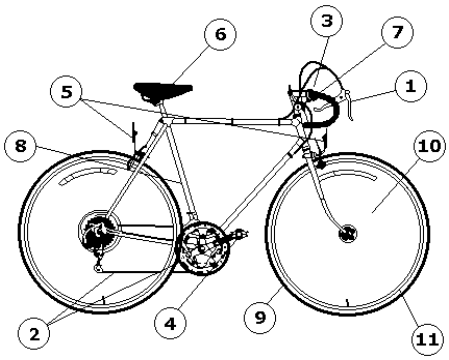
Ví dụ: Gọi {C_1, C_2, C_3, C_4, C_5} là tên của 5 môn học mà một sinh viên bắt buộc phải hoàn thành trong khóa học ngắn hạn nào đó.

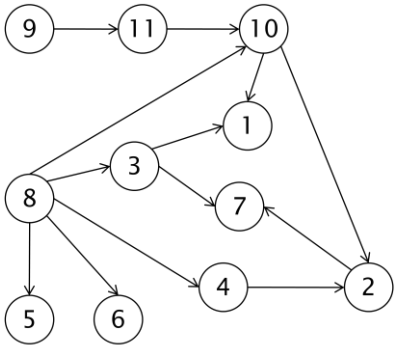


Sắp xếp topology

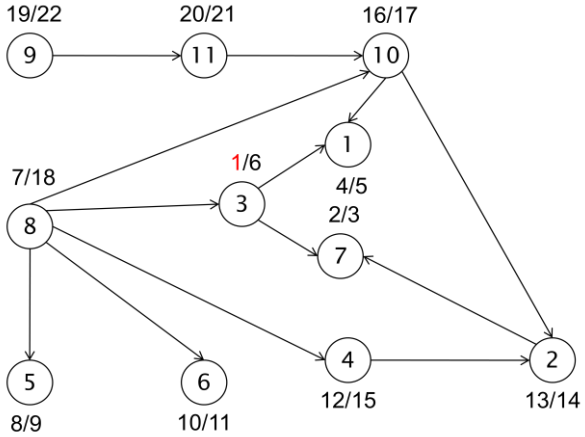


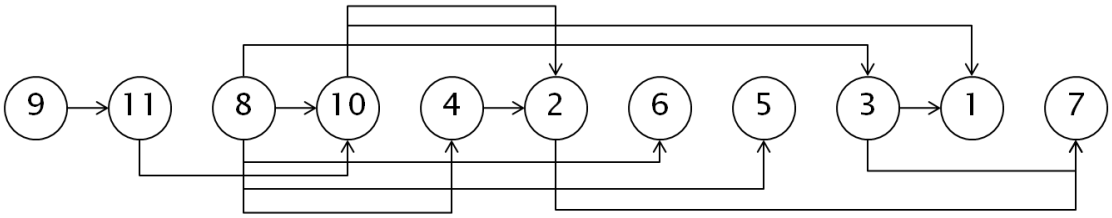
Ví dụ: Lắp ráp xe đạp





1. Lắp thắng vào tay lái
2. Gắn bộ truyền động
3. Gắn tay lái
4. Lắp bàn đạp và đĩa
5. Lắp đèn xe
6. Lắp yên xe
7. Lắp hộp điều chỉnh tốc độ
8. Lắp (tạo) khung xe
9. Lắp vỏ xe vào vành xe
10. Gắn bánh xe vào khung xe
11. Lắp vành xe vào bánh xe (đúc)





Giải thuật:

```
Topologicalsort(Graph G) {
   indegree[1 .. |V|] = {0};
   priorQueue  Q = ∅;
   for (mỗi u ∈ V)
      for (mỗi đỉnh v kề với u)
         indegree[v] ++;
   for (mỗi v ∈ V)
      if (indegree[v] == 0)
         enQueue(Q, v);
   while (!isEmpty(Q)) {
      Vertex u = deQueue(Q);          Output(u);
      for (mỗi đỉnh v kề với u)
         if( -- indegree[v] == 0 )
            enQueue(Q, v);
   }
}
```

Các giải thuật phát sinh tổ hợp

Phát sinh các hoán vị

Ví dụ:

Tập có 1 phần tử: 1

Tập có 2 phần tử: 1**2** **2**1 (*chèn từ phải sang trái dãy* 1)

Tập có 3 phần tử: 12**3** 1**3**2 **3**12 (*chèn từ phải sang trái dãy* 1**2**)

321 2**3**1 21**3** (*đảo chiều, chèn từ trái sang dãy* **2**1)

Tập có 4 phần tử: 123**4** 12**4**3 1**4**23 **4**123 (*chèn từ phải sang trái dãy* 12**3**)

4132 1**4**32 13**4**2 132**4** (*đảo chiều, chèn từ trái sang dãy* 1**3**2)

312**4** 31**4**2 3**4**12 **4**312 (*đảo chiều, chèn từ phải sang dãy* **3**12)

4321 3**4**21 32**4**1 321**4** (*đảo chiều, chèn từ trái sang dãy* **3**21)

231**4** 23**4**1 2**4**31 **4**231 (*đảo chiều, chèn từ phải sang dãy* 2**3**1)

4213 2**4**13 21**4**3 213**4** (*đảo chiều, chèn từ trái sang dãy* 21**3**)

Giải thuật Steinhause-Johnson-Trotter

Bước 1: Khởi tạo dãy phần tử tăng dần với mọi mũi tên chỉ sang trái $\overleftarrow{1}\,\overleftarrow{2}\ldots\overleftarrow{n}$.

Bước 2: Đưa dãy có được vào danh sách kết quả.

Nếu vẫn tồn tại phần tử di động trong dãy thì chuyển sang *Bước 3*.

Nếu không: Kết thúc.

Bước 3: Thực hiện ba bước dưới đây. Sau đó chuyển về *Bước 2*.

Bước 3.1: Tìm phần tử di động lớn nhất *k*.

Bước 3.2: Hoán vị *k* và phần tử kế bên được *k* chỉ đến (vẫn giữ thông tin chỉ hướng của mỗi phần tử).

Bước 3.3: Đảo hướng tất cả các phần tử có giá trị lớn hơn *k*.

Ví dụ: Dãy có 3 phần tử

1. $\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{3}$	3. $\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{2} \to \overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{2}$	5. $\overleftarrow{2}\,\overrightarrow{3}\,\overleftarrow{1}$
2. $\overleftarrow{1}\,\overrightarrow{3}\,\overleftarrow{2}$	4. $\overrightarrow{3}\,\overleftarrow{2}\,\overleftarrow{1} \to \overrightarrow{3}\,\overleftarrow{2}\,\overleftarrow{1} \to \overrightarrow{3}\,\overleftarrow{2}\,\overleftarrow{1}$	6. $\overrightarrow{2}\,\overrightarrow{1}\,\overleftarrow{3} \to \overrightarrow{2}\,\overrightarrow{1}\,\overleftarrow{3}$

Ví dụ: Dãy có 4 phần tử

1. $\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{3}\,\overleftarrow{4}$	9. $\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{4} \to \overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{4} \to \overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{4}$	17. $\overleftarrow{2}\,\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{4} \to \overleftarrow{2}\,\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{4} \to \overleftarrow{2}\,\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{4}$
2. $\overleftarrow{1}\,\overleftarrow{2}\,\overrightarrow{4}\,\overleftarrow{3}$	10. $\overleftarrow{3}\,\overleftarrow{1}\,\overrightarrow{4}\,\overleftarrow{2}$	18. $\overleftarrow{2}\,\overleftarrow{3}\,\overrightarrow{4}\,\overleftarrow{1}$
3. $\overleftarrow{1}\,\overrightarrow{4}\,\overleftarrow{2}\,\overleftarrow{3}$	11. $\overleftarrow{3}\,\overrightarrow{4}\,\overleftarrow{1}\,\overleftarrow{2}$	19. $\overleftarrow{2}\,\overrightarrow{4}\,\overleftarrow{3}\,\overleftarrow{1}$
4. $\overrightarrow{4}\,\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{3} \to \overrightarrow{4}\,\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{3} \to \overrightarrow{4}\,\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{3}$	12. $\overrightarrow{4}\,\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{2} \to \overrightarrow{4}\,\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{2} \to \overrightarrow{4}\,\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{2}$	20. $\overrightarrow{4}\,\overleftarrow{2}\,\overleftarrow{3}\,\overleftarrow{1} \to \overrightarrow{4}\,\overleftarrow{2}\,\overleftarrow{3}\,\overleftarrow{1} \to \overrightarrow{4}\,\overleftarrow{2}\,\overleftarrow{3}\,\overleftarrow{1}$
5. $\overleftarrow{4}\,\overrightarrow{1}\,\overleftarrow{3}\,\overleftarrow{2} \to \overleftarrow{4}\,\overleftarrow{1}\,\overleftarrow{3}\,\overleftarrow{2} \to \overrightarrow{4}\,\overleftarrow{1}\,\overleftarrow{3}\,\overleftarrow{2}$	13. $\overleftarrow{4}\,\overleftarrow{3}\,\overrightarrow{2}\,\overleftarrow{1} \to \overleftarrow{4}\,\overleftarrow{3}\,\overleftarrow{2}\,\overleftarrow{1} \to \overrightarrow{4}\,\overleftarrow{3}\,\overleftarrow{2}\,\overleftarrow{1}$	21. $\overrightarrow{4}\,\overrightarrow{2}\,\overleftarrow{1}\,\overleftarrow{3} \to \overrightarrow{4}\,\overrightarrow{2}\,\overleftarrow{1}\,\overleftarrow{3} \to \overrightarrow{4}\,\overrightarrow{2}\,\overleftarrow{1}\,\overleftarrow{3}$
6. $\overleftarrow{1}\,\overrightarrow{4}\,\overleftarrow{3}\,\overleftarrow{2}$	14. $\overleftarrow{3}\,\overrightarrow{4}\,\overleftarrow{2}\,\overleftarrow{1}$	22. $\overleftarrow{2}\,\overrightarrow{4}\,\overleftarrow{1}\,\overleftarrow{3}$
7. $\overleftarrow{1}\,\overleftarrow{3}\,\overrightarrow{4}\,\overleftarrow{2}$	15. $\overleftarrow{3}\,\overleftarrow{2}\,\overrightarrow{4}\,\overleftarrow{1}$	23. $\overleftarrow{2}\,\overleftarrow{1}\,\overrightarrow{4}\,\overleftarrow{3}$
8. $\overleftarrow{1}\,\overleftarrow{3}\,\overrightarrow{2}\,\overleftarrow{4} \to \overleftarrow{1}\,\overleftarrow{3}\,\overleftarrow{2}\,\overleftarrow{4} \to \overleftarrow{1}\,\overleftarrow{3}\,\overleftarrow{2}\,\overleftarrow{4}$	16. $\overleftarrow{3}\,\overleftarrow{2}\,\overleftarrow{1}\,\overrightarrow{4} \to \overleftarrow{3}\,\overleftarrow{2}\,\overleftarrow{1}\,\overleftarrow{4} \to \overleftarrow{3}\,\overleftarrow{2}\,\overleftarrow{1}\,\overleftarrow{4}$	24. $\overleftarrow{2}\,\overleftarrow{1}\,\overleftarrow{3}\,\overrightarrow{4} \to \overleftarrow{2}\,\overleftarrow{1}\,\overleftarrow{3}\,\overleftarrow{4}$

Chương trình

```
#define N 4
int  a[N + 2], pos[N + 2], dir[N + 2];
void Print() {
   for (int i = 1; i <= N; ++i)
      printf("%5d", a[i]);
}
void Move(int x, int d) {
   int z = a[pos[x] + d];

   a[pos[x]] = z;
   a[pos[x] + d] = x;

   pos[z] = pos[x];
   pos[x] = pos[x] + d;
}
void SJT(int n) {
   if (n > N)
      Print();
   else {
      SJT(n + 1);
      for(int i = 1; i <= n - 1; ++i) {
         Move(n, dir[n]);
         SJT(n + 1);
      }
      dir[n] = -dir[n];
   }
}
void main () {
   for (int i = 1; i <= N; ++i) {
      dir[i] = -1;
      a[i] = i;
      pos[i] = i;
   }
   SJT(1);
}
```

Thứ tự từ điển học (lexicographic order)

Ý tưởng: Quét dãy hoán vị hiện tại từ phải sang trái để tìm cặp phần tử liên tiếp đầu tiên a_i và a_{i+1} sao cho $a_i < a_{i+1}$. Khi đó, tìm phần tử nhỏ nhất trong các số từ a_{i+1} đến a_n nhưng phải lớn hơn a_i. Đặt số này vào vị trí i, những con số còn lại (từ vị trí i đến n) sẽ được đặt vào các vị trí từ $i + 1$ đến n theo thứ tự tăng dần.

Giải thuật (Hàm tìm dãy hoán vị kế tiếp dãy được truyền vào)

```
NextPerm(a[1 .. n]) {
   k = n - 1;
   while (a[k] > a[k+1]) {
      k--;
      if (k == 0)
         return;
   }
   i = n;
   while (a[k] > a[i])
      i--;
   a[k] ⇆ a[i];
   r = n;
   s = k + 1;
   while (r > s) {
      a[r] ⇆ a[s];
      r--;
      s++;
   }
}
```

Phát sinh tập con

Ví dụ: Phát sinh tập lũy thừa của tập $\{a_1, a_2, a_3\}$

n	Các tập con							
0	$\varnothing$							
1	$\varnothing$	$\{a_1\}$						
2	$\varnothing$	$\{a_1\}$	$\{a_2\}$	$\{a_1, a_2\}$				
3	$\varnothing$	$\{a_1\}$	$\{a_2\}$	$\{a_1, a_2\}$	$\{a_3\}$	$\{a_1, a_3\}$	$\{a_2, a_3\}$	$\{a_1, a_2, a_3\}$

Chương trình

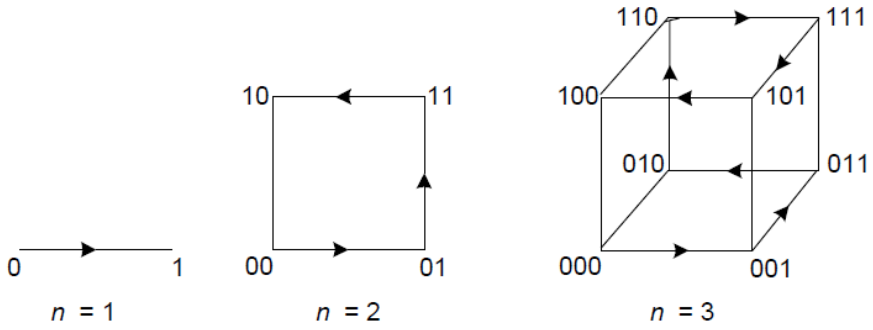
```
void Subsets(int n, int a[]) {
   if (n == N)
      Print(a);
   else {
      a[n] = 0;   Subsets(n + 1, a);
      a[n] = 1;   Subsets(n + 1, a);
   }
}
Subsets(0, a);
```

Thứ tự *nén* (*squashed order*)

Chương trình

```
void Subsets(int n, int a[]) {
   if (n == 0)
      PrintIt(a);
   else {
      a[n - 1] = 0;
      Subsets(n - 1, a);
      a[n - 1] = 1;
      Subsets(n - 1, a);
   }
}
Subsets(N, a);
```

(*binary reflected*) ***Gray code***



Qui luật để hình thành *Gray code* chiều dài *n* là:

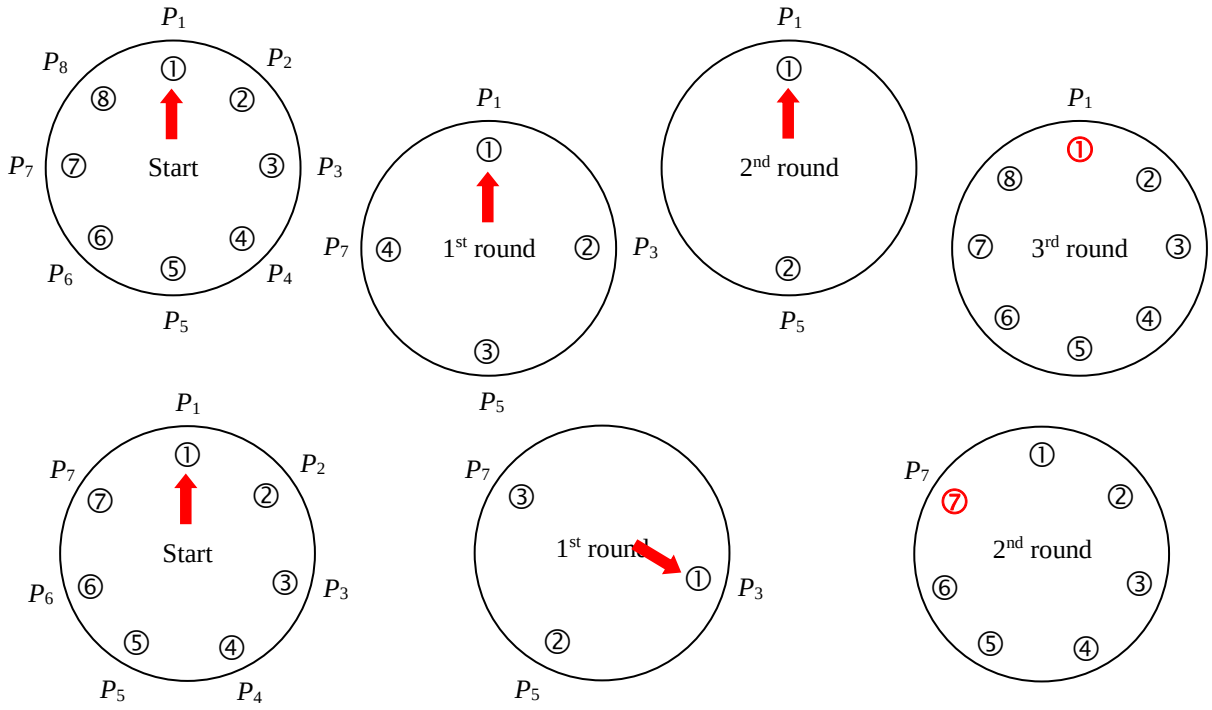
(1) Tạo *Gray code* chiều dài *n* – 1.
(2) Nhân bản lên thành hai.
(3) Thêm số 0 vào đầu các chuỗi bit của phiên bản nhân bản một và thêm số 1 vào đầu các chuỗi bit của phiên bản nhân bản hai.
(4) Đảo thứ tự các chuỗi bit của phiên bản nhân bản hai và ghép vào đuôi của phiên bản nhân bản một.

Kỹ thuật giảm với tỉ lệ không đổi

Bài toán Josephus

Phát biểu: Có *n* người đứng thành vòng tròn. Bắt đầu từ người có số thứ tự 1, người kế bên sẽ bị loại bỏ cho đến khi chỉ còn một người. *Hãy xác định số thứ tự J(n) ban đầu của người sẽ trụ lại cuối cùng.*

Ví dụ: $J(8) = 1$, $J(7) = 7$, $J(6) = 5$.



Kỹ thuật giảm với lượng biến đổi

Tìm USCLN(a, b) – Giải thuật Euclid

Phát biểu: Gọi $a, b \in \mathbb{Z}^+$. Tìm USCLN của hai số này.

Giải thuật

```
gcd(a, b) {
   while (b) {
      t = b;
      b = a % b;
      a = t;
   }
   return   a;
}
```

Cây nhị phân tìm kiếm

Phát biểu: Xác định sự hiện diện của giá trị *k* trên cây nhị phân tìm kiếm.

Vấn đề chọn (Selection problem)

Phát biểu: Tìm phần tử *nhỏ thứ k* trong dãy *n* phần tử, với $k \in [1, n]$.

Giải thuật

```
Selection(S[], lower, upper, k) {
   index = random(lower, upper);
   pos = Partition(S, lower, upper, index);      // S[pos] = pivot
   if (pos == k)
      return S[pos];
   if (pos > k)
      Selection(S, lower, pos - 1, k);
   if (pos < k)
      Selection(S, pos + 1, upper, k);
}
Partition(S, lower, upper, pos) {
   pivot = S[pos];
   S[lower] ⇆ S[pos];
   pos = lower;
   for (i = lower + 1; i ≤ upper; i++)
      if (pivot > S[i]) {
         pos++;
         S[i] ⇆ S[pos];
      }
   S[lower] ⇆ S[pos];
   return   pos;
}
```

Tìm kiếm nội suy