TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN MÔN TỔ HỢP & ĐỒ THỊ

10 bài lab

Người hướng dẫn: **Thầy TRẦN LƯƠNG QUỐC ĐẠI**

Người thực hiện: **Thái Kim Thư – 51800816**

Khoá : **22**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN MÔN TỔ HỢP & ĐỒ THỊ

10 bài lab

Người hướng dẫn: **Thầy TRẦN LƯƠNG QUỐC ĐẠI**

Người thực hiện: **Thái Kim Thư – 51800816**

Khoá : **22**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021

LỜI CẢM ƠN

Đây là phần tác giả **tự viết** ngắn gọn, thể hiện sự biết ơn của mình đối với những người đã giúp mình hoàn thành Luận văn/Luận án. Tuyệt đối không sao chép theo mẫu những "lời cảm ơn" đã có.

TÓM TẮT

Trình bày tóm tắt vấn đề nghiên cứu, các hướng tiếp cận, cách giải quyết vấn đề và một số kết quả đạt được, những phát hiện cơ bản trong vòng 1 -2 trang.

MỤC LỤC

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

CÁC KÝ HIỆU

f *Tần số của dòng điện và điện áp (Hz)*

p *Mật độ điện tích khối (C/m3)*

CÁC CHỮ VIẾT TẮT

CSTD	Công suất tác dụng
MF	Máy phát điện
BER	Tỷ lệ bít lỗi

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

DANH MỤC BẢNG

CHƯƠNG 1 – TỔ HỢP

1.1 Mixed radix generation (Tạo cơ số hỗn hợp).

1.1.1 Giới thiệu bài toán.

1.1.2 Ý tưởng thuật toán.

- Input:
- Output:

1.1.3 Các bước thực hiện.

1.1.4 Mã giả.

Algorithm 1 Mixed radix generation (**M**)

Auxiliary variables a_0 and m_0 are introduced for convenience.

Step 1: [Initialize] Set $a_j \longleftarrow 0$ for $0 \leq j \leq n$, and set $m_0 \longleftarrow 2$

Step 2: [Visit] Visit the n-tuple $(a_1, ..., a_n)$. (The program that wants to examine all n-tuples now does its thing.)

Step 3: [Prepare to add one] Set $j \longleftarrow n$

Step 4: [Carry if necessary] If $a_j = m_j - 1$, set $a_j \longleftarrow 0$, $j \longleftarrow j - 1$, and repeat this step.

Step 5: [Increase, unless done] If $j = 0$, terminate the algorithm. Otherwise set $a_j \longleftarrow a_j + 1$ and go back to **Step 2**.

Hình 1: Mã giả Mixed radix generation.

1.1.5 Mã nguồn.

```python
def MixedRadixGeneration (m1):
    a = [0] * (len(m1)+1)
    m = [0] + m1
    n = len(m1)
    flag = True
    while flag == True:
        print(a[1:])
        j = n
        while a[j] == m[j] - 1:
            a[j] = 0
            j = j - 1
        if j == 0:
            flag = False
        else:
            a[j] += 1


m = [1,3,2]
MixedRadixGeneration(m)
```

Hình 2: Mã nguồn Mixed radix generation.

1.2 Heap algorithm (Thuật toán đống).

1.2.1 Giới thiệu bài toán.

1.2.2 Ý tưởng thuật toán.

1.2.3 Các bước thực hiện.

1.2.4 Mã giả.

Algorithm 1 Heap algorithm

Input: n: integer, A: an array

```
if n = 1 then output A
else
  for i:=0; i¡ n; i ++ do
    If n is even then
      Swap(A[i], A[n-1])
    else
      Swap(A[0], A[n-1])
```

Output: the sets is permutated

Hình 3: Mã giả Heap algorithm.

1.2.5 Mã nguồn.

```python
def heapPer (a, n):
    if n == 1:
        print(a)
        return
    for i in range(0,n):
        heapPer(a,n-1)
        if n % 2 == 0:
            temp = a[i]
            a[i] = a[n-1]
            a[n-1] = temp
        else:
            temp = a[0]
            a[0] = a[n-1]
            a[n-1] = temp

a = [1,2,3]
n = 3
heapPer(a,n)
```

Hình 4: Mã nguồn Heap algorithm.

1.3 Steinhaus Johnson Trotter.

1.3.1 Giới thiệu bái toán.

1.3.2 Ý tưởng thuật toán.

1.3.3 Các bước thực hiện.

1.3.4 Mã giả.

Algorithm 1 Steinhaus–Johnson–Trotter

Input: given k is current mobile integer, p[]: input sequence, pi[]: invert of p, d[] is direction of mobile integer.

```
begin
    if k ≤ N then
      JohnsonSteinhausTrotter (k+1, p, pi, d)
      for c := 1...k-1 loop
        x := pi[k];
        y := x + d[k];
        j := p[y];
        p[x] := j; pi[j] := x;
        p[y] := k; pi[k] := y;
        JohnsonSteinhausTrotter (k+1, p, pi, d)
      end loop
      d[k] := -d[k]
    else
      output(p)
    end if
end
```

Hình 5: Mã giả Heap algorithm.

1.3.5 Mã nguồn.

```
def JohnsonSteinhausTrotter(k,p,pi,d):
    if k <= len(p):
        JohnsonSteinhausTrotter(k+1, p, pi, d)
        for c in range(0,k):
            x = pi[k]
            y = x + d[k]
            j = p[y]
            p[x] = j
            pi[j] = x
            p[y] = k
            pi[k] = y
            JohnsonSteinhausTrotter(k+1, p, pi, d)
        d[k] = -d[k]
    else:
        print(p)

k = 1
p = [1,2,3]
pi = [3,2,1]
d = [-1,-1,-1]
JohnsonSteinhausTrotter(k,p,pi,d)
```

Hình 6: Mã nguồn Steinhaus Johnson Trotter.

CHƯƠNG 2 – ĐỒ THỊ