BÀI TẬP CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT 2

Chương 1

Bài 1 Chứng minh các mệnh đề sau:
a. Nếu t(n) = O(g(n)) thì g(n) = Ω(t(n)).
b. Θ(αg(n)) = Θ(g(n)), với α >0.
Bài 2 Viết và đánh giá độ phức tạp của các thuật giải đệ qui và không đệ qui tìm kiếm nhị phân trên một dãy số thực đã được sắp xếp.
Bài 3 Viết thuật giải tính 2^n theo hệ thức $2^n = 2^{n-1} + 2^{n-1}$. Tính toán thời gian chạy của thuật giải đã viết.
Bài 4 Viết các giải thuật đệ qui và không đệ qui tính lũy thừa a^n (a là số thực, n là số nguyên dương), đánh giá độ phức tạp các giải thuật đã viết. Nếu các thuật giải đã viết có độ phức tạp là O(n) thì hãy viết thuật giải tốt hơn.
Bài 5 Xét thuật giải sau.
Riddle($A[0..n-1]$)   // $A[0..n-1]$ là một mảng các số thực
  if n == 1 **return** $A[0]$
  else temp = Riddle(A[0..n − 2])
    if temp ≤ A[n − 1] **return** temp
    else return A[n − 1]
a. Hãy cho biết thuật giải tính toán gì?
b. Phân tích độ phức tạp thuật giải.
Bài 6 Viết các hàm C/C++ tính 2^n theo các hệ thức $2^n = 2^{n-1} + 2^{n-1}$ và $2^n = 2^{n-1}.2$, so sánh thời gian chạy thực tế của chúng ứng với n=100, 1000, 10.000. 100.000.
Bài 7 Viết các hàm C/C++ đệ qui và không đệ qui tính giá trị của a_n theo n là số tự nhiên, được định nghĩa bởi hệ thức $a_n=a_{n-1}+ a_{n-1}$ với $a_0 =1$, $a_1=1$. So sánh thời gian chạy thực tế của chúng ứng với n=20, 40, 80, 90, 100.

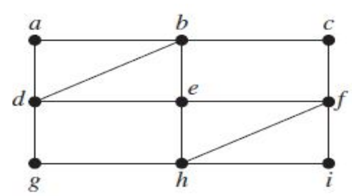
Chương 2

Bài 1 Tính số phần tử nhiều, ít nhất của một heap nhị phân có chiều cao h.
Bài 2 Chứng minh một heap nhị phân n phần tử thì có chiều cao là $\lfloor \log_2 n \rfloor$.
Bài 3 Biểu diễn quá trình thực hiện thủ tục (thuật giải) BuildMaxHeap trên mảng A = ⟨5, 3, 17, 10, 84, 19, 6, 22, 9⟩.
Bài 4 Biểu diễn quá trình thực hiện HeapSort trên mảng A = ⟨5, 3, 17, 10, 84, 19, 6, 22, 9⟩.
Bài 5 Viết thuật giải HeapSort để sắp xếp một mảng A gồm n số nguyên theo thứ tự giảm dần.
Bài 6 Biểu diễn quá trình thực thi thủ tục Partition trên mảng A = ⟨13, 3, 9, 5, 12, 8, 7, 4, 11, 2, 6⟩.
Bài 7 Biểu diễn quá trình thực hiện thuật giải Quicksort trên mảng A = ⟨13, 3, 9, 5, 12, 8, 7, 4, 11, 2, 6⟩.
Bài 8 Viết thuật giải Quicksort để sắp xếp một mảng A gồm n số nguyên theo thứ tự giảm dần.
Bài 9 Đánh giá độ phức tạp của thuật giải QuickSort trong trường hợp mảng đầu vào có thứ tự tăng hoặc giảm dần.
Bài 10 Biểu diễn quá trình thực hiện thuật giải CountingSort trên mảng A = ⟨6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2⟩.

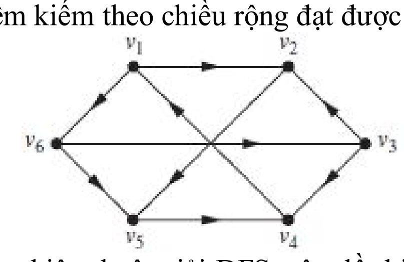
Bài 11 Viết thuật giải CountingSort để sắp xếp một mảng n số nguyên theo thứ tự giảm dần.
Bài 12 Biểu diễn quá trình thực hiện thuật giải BucketSort trên mảng A = ⟨.79, .13, .16, .64, .39, 20, .89, .53, .71, .42⟩.
Bài 13 Viết một thuật giải mở rộng của thuật giả**i** BucketSort để có thể sắp xếp một mảng n số thực không âm bất kỳ.
Bài 14 Viết các chương trình C/C++ hiện thực thuật giải HeapSort để sắp xếp một mảng n số thực theo thứ tự tăng (giảm) dần.
Bài 15 Viết các chương trình C/C++ hiện thực thuật giải QuickSort để sắp xếp một mảng n số thực theo thứ tự tăng (giảm) dần.
Bài 16 Giả sử mỗi sinh viên được xác định bởi mã số, họ, tên và tuổi. Hãy sử dụng thuật giải Heapsort viết một chương trình C/C++ để sắp xếp n sinh viên theo thứ tự tăng dần theo tên của sinh viên.
Bài 17 Hãy định nghĩa quan hệ thứ tự trên các phân số dạng a/b (a và b là các số nguyên, b≠0). Sử dụng thuật giải QuickSort viết một chương trình C/C++ để sắp xếp một mảng n phân số theo thứ tự giảm dần.
Bài 18 Viết hàm C/C++ hiện thực thuật giải CountingSort để sắp xếp một mảng n số nguyên nhỏ hơn một số k cho trước.
Bài 19 Viết hàm C/C++ hiện thực thuật giải BucketSort để sắp xếp một mảng n số thực thực không âm nhỏ hơn 1.
Bài 20 Sử dụng BucketSort, viết hàm C/C++ để sắp xếp một mảng n số thực thực không âm (có thời gian chạy tương đương với BucketSort).

Chương 3

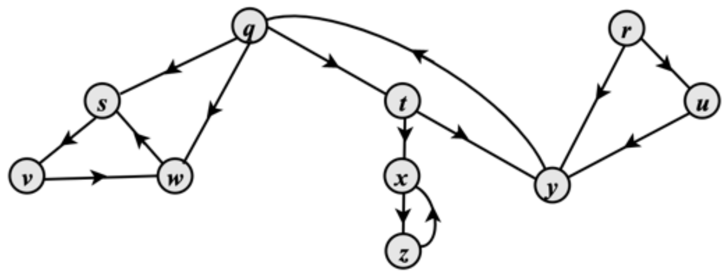
Bài 1 Viết thuật giải tìm kiếm theo chiều rộng BFS bắt đầu từ một đỉnh s của đồ thị G. Biểu diễn quá trình thực hiện thuật giải BFS trên đồ thị vô hướng sau với đỉnh bắt đầu tìm kiếm là *b*. Vẽ cây tìm kiếm kiếm theo chiều rộng đạt được sau khi thực hiện thuật giải.



Bài 2 Biểu diễn quá trình thực hiện thuật giải BFS trên đồ thị có hướng sau với đỉnh bắt đầu tìm kiếm là v_2. Vẽ cây tìm kiếm kiếm theo chiều rộng đạt được sau khi thực hiện thuật giải.



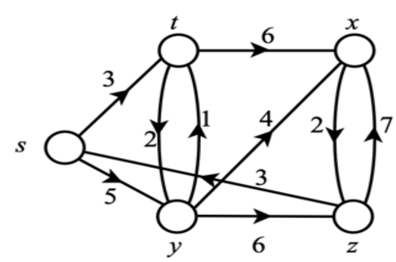
Bài 3 Biểu diễn quá trình thực hiện thuật giải DFS trên đồ thị vô hướng trong bài 1. Vẽ cây (rừng) tìm kiếm kiếm theo chiều sâu đạt được sau khi thực hiện thuật giải.
Bài 4 Viết thuật giải tìm kiếm theo chiều rộng DFS trên đồ thị G. Biểu diễn quá trình thực hiện thuật giải DFS trên đồ thị có hướng sau với giả sử trong quá trình thực thi DFS tìm kiếm các đỉnh theo thứ tự alphabet của tên đỉnh. Chỉ ra thời gian phát hiện và kết thúc tìm kiếm mỗi đỉnh của đồ thị.



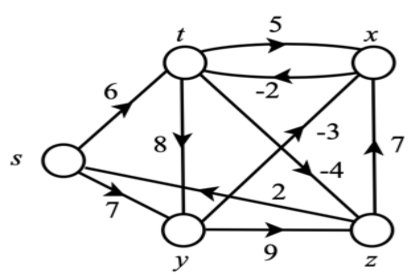
Bài 5 Viết thuật giải kiểm tra xem đồ thị vô hướng G có chu trình đi qua đỉnh s hay không.
Bài 6 Viết thuật giải kiểm tra xem đỉnh s và đỉnh t có thuộc cùng một thành phần liên thông của đồ thị vô hướng G hay không.
Bài 7 Biểu diễn đồ thị như ma trận hoặc danh sách kề và viết các chương trình C/C++ hiện thực các giải thuật DFS và BFS.
Bài 8 Viết chương trình xác định xem một đồ thị vô hướng có liên thông hay không.
Bài 9 Viết chương trình tính số thành phần liên thông của một đồ thị vô hướng.
Bài 10 Viết chương trình kiểm tra xem đồ thị vô hướng G có chu trình đi qua đỉnh s hay không.
Bài 11 Viết chương trình kiểm tra xem đỉnh s và đỉnh t có thuộc cùng một thành phần liên thông của đồ thị vô hướng G hay không.

Chương 4

Bài 1 Chỉ ra hai cây các đường đi ngắn nhất trên đồ thị sau.

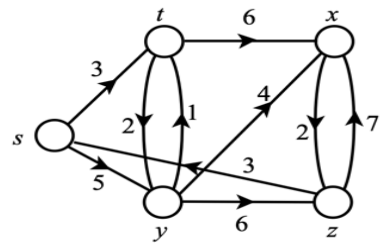


Bài 2 Viết thuật giải Bellman-Ford tìm đường đi ngắn nhất từ một đỉnh s đến các đỉnh còn lại của đồ thị có trọng số là các số thực bất kỳ. Cho đồ thị có hướng, có trọng số như sau.

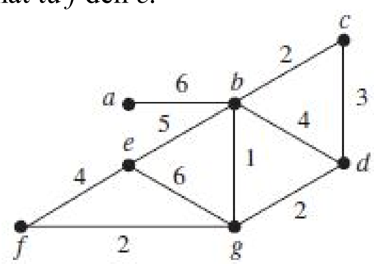


a. Biểu diễn quá trình thực hiện thuật giải Bellman-Ford trên đồ thị với đỉnh bắt đầu tìm các đường đi ngắn nhất là *z*. Trong mỗi chuyến Relax các cạnh theo thứ tự (*t*, *x*), (*t*, *y*), (*t*, *z*), (*x*, *t*), (*y*, *x*), (*y*, *z*), (*z*, *x*), (*z*, *s*), (*s*, *t*), (*s*, *y*) và chỉ ra các giá trị của *d* và π của mỗi đỉnh sau mỗi chuyến.
b. Thay đổi trọng số cạnh (*z*, *x*) thành 4. Thực hiện thuật giải Bellman-Ford bắt đầu từ đỉnh *s*.

Bài 3 Viết thuật giải Dijkstra tìm đường đi ngắn nhất từ một đỉnh s đến các đỉnh còn lại của đồ thị có trọng số. Biểu diễn quá trình thực hiện thuật giải Dijkstra tìm các đường đi ngắn nhất từ đỉnh *t* đến các đỉnh còn lại trên đồ thị sau. Trong mỗi bước tính toán chỉ ra các giá trị d[v], π[v] của các đỉnh v và tập *S* = {v | d[v] = d(t, v)}. Cho biết đường đi và độ dài của đường đi ngắn nhất từ t đến s.



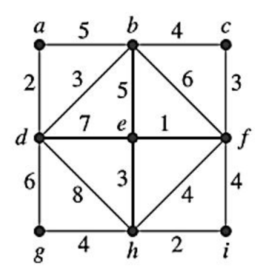
Bài 4 Biểu diễn quá trình thực hiện thuật giải Dijkstra tìm các đường đi ngắn nhất từ đỉnh *f* đến các đỉnh còn lại trên đồ thị sau. Trong mỗi bước tính toán chỉ ra các giá trị d[v], π[v] của các đỉnh v và tập *S* = {v | d[v] = d(f, v)}. Cho biết cây đường đi ngắn nhất. Cho biết đường đi và độ dài của đường đi ngắn nhất từ *f* đến *c*.



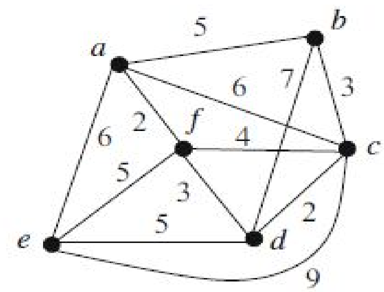
Bài 5 Viết chương trình hiện thực thuật giải Dijkstra để tìm các đường đi ngắn nhất từ một đỉnh s đến các đỉnh còn lại trên một đồ thị có trọng số.
Bài 6 Viết chương trình hiện thực thuật **giải Bellman-Ford** để tìm các đường đi ngắn nhất từ một đỉnh s đến các đỉnh còn lại trên một đồ thị có trọng số.

Chương 5

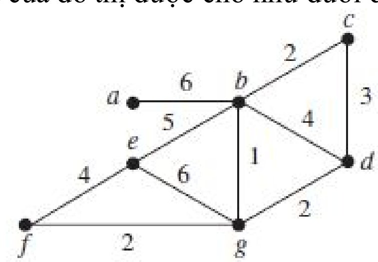
Bài 1 Cho *G* = (*V*, *E*) là một đồ thị vô hướng có số cạnh bằng số đỉnh. Chứng minh rằng *G* có ít nhất một chu trình.
Bài 2 *G* là một rừng gồm *k* cây, *n* đỉnh. Hãy tính số cạnh của *G*.
Bài 3 Cho *G* = (*V*, *E*) là một đồ thị vô hướng liên thông có trọng số. Giả sử *e* = (*u*, *v*) ∈ *E* là cạnh có trọng số bé nhất trong *G*. Chứng minh rằng có một cây bao trùm bé nhất chứa cạnh *e*.
Bài 4 Viết thuật giải Kruskal để tìm cây bao trùm bé nhất của đồ thị vô hướng liên thông có trọng số. Biểu diễn các bước thực hiện thuật giải Kruskal để tìm cây bao trùm bé nhất của các đồ thị được cho dưới đây.



Bài 5 Viết thuật giải Prim để tìm cây bao trùm bé nhất của đồ thị vô hướng liên thông có trọng số bắt đầu từ một đỉnh s. Biểu diễn các bước thực hiện thuật giải Prim bắt đầu từ đỉnh *e* để tìm cây bao trùm bé nhất của đồ thị được cho như sau.



Bài 6 Bài Biểu diễn quá trình thực hiện thuật giải Kruskal và thuật giải Prim (bắt đầu từ đỉnh *c*) để tìm cây bao trùm bé nhất của đồ thị được cho như dưới đây.



Bài 7 Viết chương trình hiện thực thuật giải Prim để tìm cây bao trùm nhỏ nhất của đồ thị vô hướng liên thông có trọng số.

Bài 8 Viết chương trình hiện thực thuật giải Kruskal để tìm cây bao trùm nhỏ nhất của đồ thị vô hướng liên thông có trọng số.