## Chương 10: Kỹ thuật Tham (Greedy Technique)

*Giới thiệu chung*

Bài toán đổi tiền xu: lối suy nghĩ "tham" luôn cho ra lời giải tối ưu với bài toán cụ thể này khi mệnh giá các đồng xu được thiết kế hợp lý.

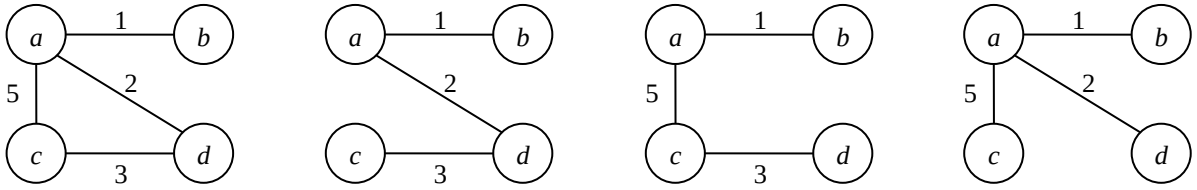
Tiếp cận này sẽ xây dựng lời giải thông qua một dãy các bước. Tại mỗi bước, chọn lựa/lời giải được đưa ra phải:

- Khả thi
- Tối ưu cục bộ
- Không thể thay đổi

*Vấn đề cây khung tối thiểu*

*Định nghĩa*: Cây khung (*spanning tree*) của đồ thị liên thông $G$ là một đồ thị con liên thông không có chu trình và chứa mọi đỉnh của $G$. Cây khung tối thiểu (*minimum spanning tree*) của đồ thị liên thông có trọng số $G_w$ là cây khung của $G_w$ có trọng số cây nhỏ nhất (trọng số cây là tổng các trọng số của các cạnh).

*Ví dụ*:

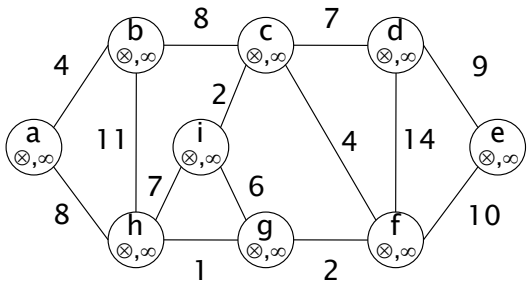



Đồ thị $G$ $w(T_1) = 6$ $w(T_2) = 9$ $w(T_3) = 8$

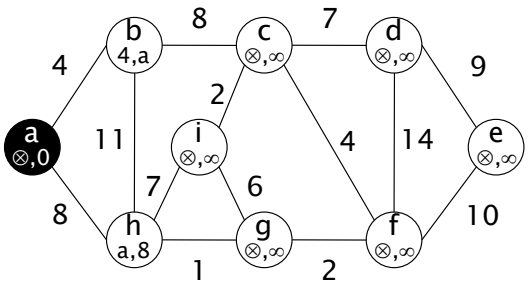
*Giải thuật (của) Prim*

*Ý tưởng*: Cây khung tối thiểu sẽ được xây dựng dần dần bằng cách ban đầu, cây chỉ là một đỉnh bất kỳ của đồ thị. Sau mỗi lần lặp, cây này sẽ mở rộng ra theo tư duy "tham" khi nối thêm vào nó một đỉnh (chưa thuộc cây) gần nhất.

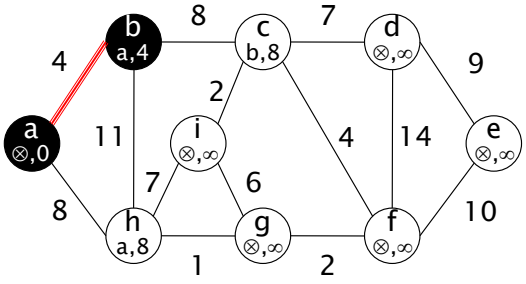
*Ví dụ*: Xét đồ thị có trọng số dưới đây và quá trình xây dựng cây khung tối thiểu của nó. Giả sử đỉnh *a* được chọn làm *root*.


a ⊗,0
b ⊗,∞
c ⊗,∞
d ⊗,∞
e ⊗,∞
f ⊗,∞
g ⊗,∞
h ⊗,∞
i ⊗,∞
4
8
8
11
7
2
6
4
14
9
10
1
2

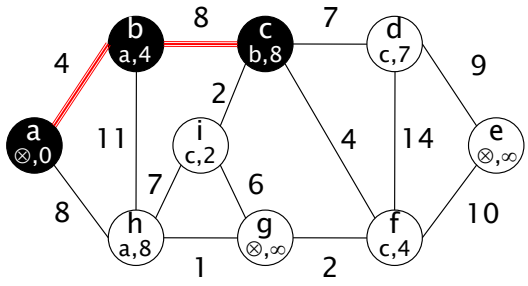
(a)


a ⊗,0
b 4,a
c ⊗,∞
d ⊗,∞
e ⊗,∞
f ⊗,∞
g ⊗,∞
h a,8
i ⊗,∞
4
8
8
11
7
2
6
4
14
9
10
1
2

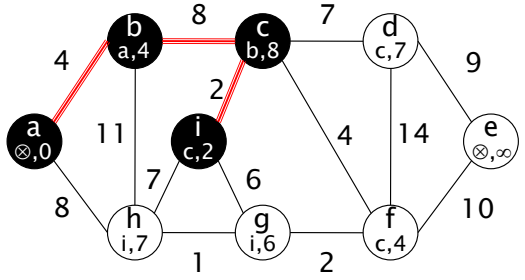
(b)


a ⊗,0
b a,4
c b,8
d ⊗,∞
e ⊗,∞
f ⊗,∞
g ⊗,∞
h a,8
i ⊗,∞
4
8
8
11
7
2
6
4
14
9
10
1
2

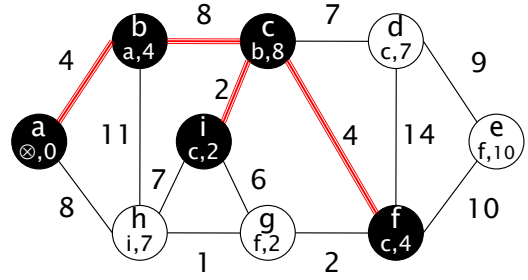
(c)


a ⊗,0
b a,4
c b,8
d c,7
e ⊗,∞
f c,4
g ⊗,∞
h a,8
i c,2
4
8
8
11
7
2
6
4
14
9
10
1
2

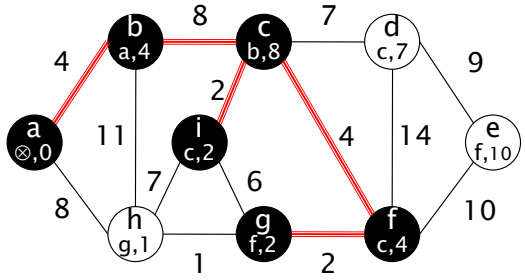
(d)


a ⊗,0
b a,4
c b,8
d c,7
e ⊗,∞
f c,4
g i,6
h i,7
i c,2
4
8
8
11
7
2
6
4
14
9
10
1
2

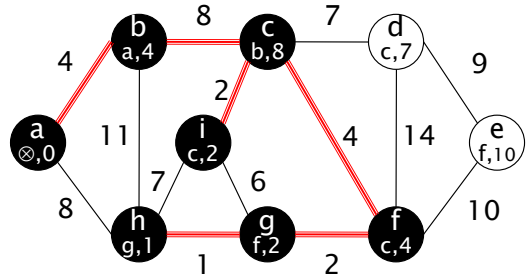
(e)


a ⊗,0
b a,4
c b,8
d c,7
e f,10
f c,4
g f,2
h i,7
i c,2
4
8
8
11
7
2
6
4
14
9
10
1
2

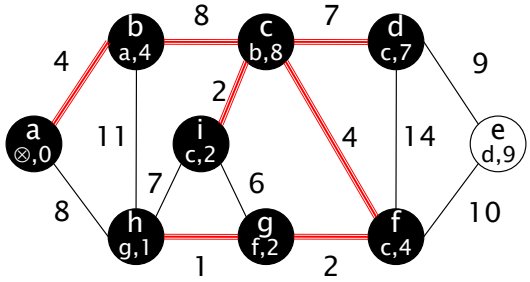
(f)


a ⊗,0
b a,4
c b,8
d c,7
e f,10
f c,4
g f,2
h g,1
i c,2
4
8
8
11
7
2
6
4
14
9
10
1
2

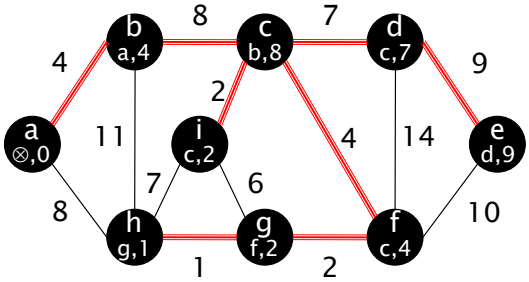
(g)


a ⊗,0
b a,4
c b,8
d c,7
e f,10
f c,4
g f,2
h g,1
i c,2
4
8
8
11
7
2
6
4
14
9
10
1
2

(h)


a ⊗,0
b a,4
c b,8
d c,7
e d,9
f c,4
g f,2
h g,1
i c,2
4
8
8
11
7
2
6
4
14
9
10
1
2

(i)


a ⊗,0
b a,4
c b,8
d c,7
e d,9
f c,4
g f,2
h g,1
i c,2
4
8
8
11
7
2
6
4
14
9
10
1
2

(j)

*Giải thuật*

```
void     MST_PRIM(G, root) {
   for (mỗi đỉnh u ∈ V) {
      prior_u = ∞;
      parent_u = NIL;
   }
   prior_root = 0;
   createQueue(Q, V);
   while (Q ≠ ∅) {
      u = extractQueue(Q);
      for (mỗi đỉnh v là lân cận của u)
         if (v ∈ Q and weight_u,v < prior_v) {
            parent_v = u;
            prior_v = weight_u,v;
            updateQueue(Q, v);
         }
   }
}
```

*Bài toán tìm đường đi ngắn nhất từ một đỉnh đến mỗi đỉnh còn lại*

*Phát biểu*: Cho đồ thị G = (E, V) liên thông có trọng số. Chọn đỉnh bất kỳ, gọi là *source*, tìm con đường ngắn nhất từ *source* đến mỗi đỉnh còn lại.

*Giải thuật (của) Dijkstra*

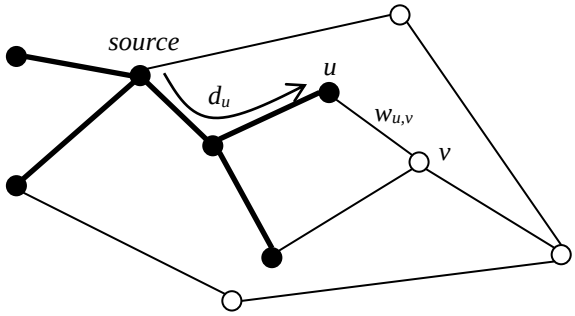
*Ý tưởng*: Giải thuật tìm con đường ngắn nhất từ *source* đến đỉnh gần nhất của nó, rồi đến đỉnh thứ hai, thứ ba, …

Gọi $T_i$ là cây con gốc *source*, chứa các đỉnh và cạnh được phát hiện sau lượt duyệt thứ *i*. Các đỉnh kề với những đỉnh trong $T_i$ hình thành nên cái gọi là những *đỉnh rìa*.

Để nhận diện đỉnh gần nhất kế tiếp, giải thuật:

– Tính, với mỗi đỉnh rìa *v*, *tổng* của (*i*) khoảng cách đến đỉnh *u* ($\in T_i$) gần nhất (chính là trọng số của cạnh (*u*, *v*)) và (*ii*) chiều dài $d_u$ của con đường ngắn nhất từ *source* đến đỉnh *u* (đã xác định).

– (và) Chọn đỉnh rìa *v* nào có *tổng* nhỏ nhất, gọi là *v**.




Về mặt kỹ thuật, mỗi đỉnh gồm hai nhãn:

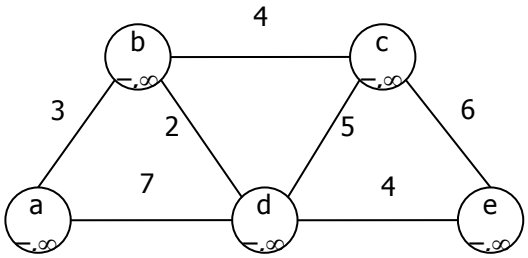
– Nhãn số: Chiều dài con đường ngắn nhất từ *source* đến đỉnh này (ban đầu là +∞).

– Nhãn chữ: Tên của đỉnh (đã thuộc về cây) gần nhất của nó (ban đầu là NIL).

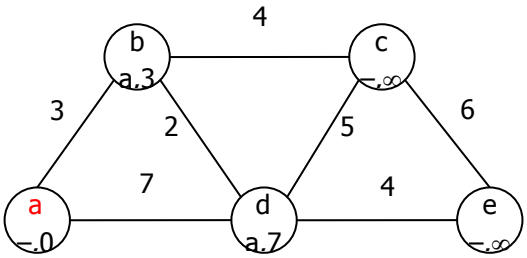
Mỗi khi nhận diện được đỉnh $v^*$, tiến hành hai thao tác sau:

– Loại $v^*$ ra khỏi nhóm đỉnh rìa và đưa vào tập của các đỉnh thuộc về cây $T_{i+1}$.

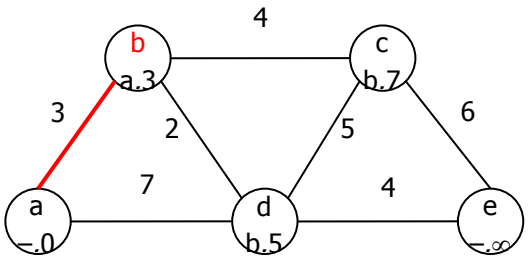
– Với mỗi đỉnh rìa $v$ còn lại kề với $v^*$ bởi cạnh có trọng số $w_{v^*,v}$, nếu $d_{v^*} + w_{v^*,v} < d_v$ thì cập nhật nhãn chữ của đỉnh $v$ là $v^*$ và nhãn số $d_v$ là giá trị $d_{v^*} + w_{v^*,v}$.



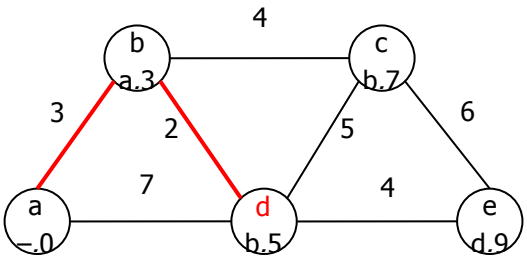

(a)



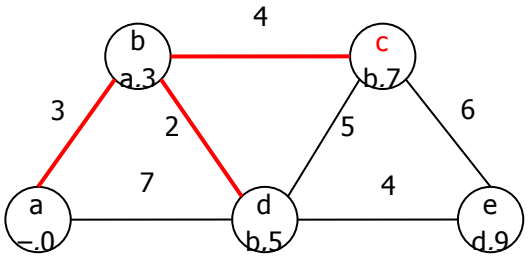

(b)



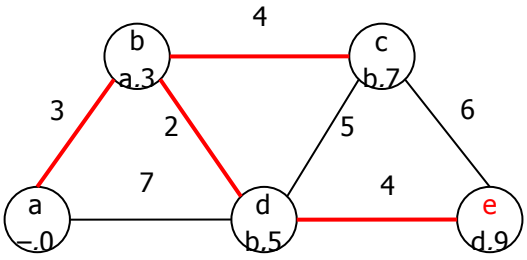

(c)




(d)




(e)




(f)

*Giải thuật*

```
Dijkstra(G(V, E), source) {
   for (mỗi đỉnh v ∈ V) {
      d_v = ∞;
      parent_v = NIL;
   }
   d_source = 0;
   createQueue(Q, V);
   T = ∅;
   while (Q ≠ ∅) {
      v* = extractQueue(Q);
      T ∪= v*;
      for (mỗi đỉnh v ∈ (V \ T) và (v*, v) ∈ E)
         if (d_v* + w_v*,v < d_v) {
            parent_v = v*;
            d_v = d_v* + w_v*,v;
            updateQueue(Q, v);
         }
   }
}
```

*Cây Huffman*

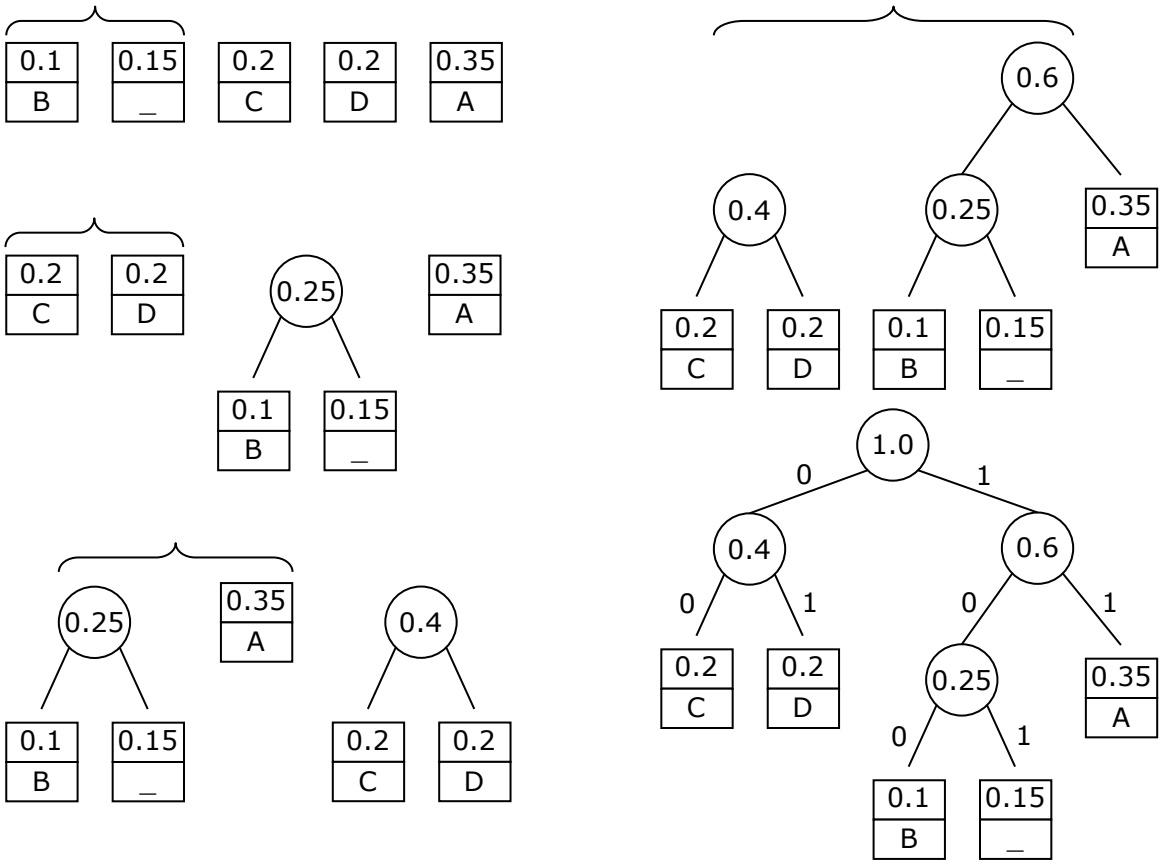
*Phát biểu*: Mã hóa một văn bản mà các ký tự xuất hiện thuộc về bảng chữ cái Σ kích thước *n*. Cách thức thực thi là thay thế mỗi ký tự bằng một chuỗi bit tương ứng, gọi là mã tự (*codeword*).

*Mã hóa chiều dài cố định* (*fixed-length encoding*)
*Mã hóa chiều dài biến đổi* (*variable-length encoding*)
*Mã phi tiền tố* (*prefix-free code*)

*Ý tưởng*

*Bước 1*: Khởi tạo *n* (= |Σ|) cây, mỗi cây chỉ có một nút với nhãn là một ký tự của bảng chữ cái Σ. Ghi nhớ tần suất sử dụng của ký tự vào nút gốc của cây và gọi đó là trọng số cây.
*Bước 2*: Nếu còn tồn tại từ hai cây con trở lên thì sang Bước 3, ngược lại thì dừng.
*Bước 3*: Tìm hai cây có trọng số nhỏ nhất và đưa chúng trở thành hai cây con trái/phải của cây nhị phân mới với nút gốc lưu tổng trọng số của hai cây con tìm được. Quay lại Bước 2.

*Ví dụ*: Σ = {A, B, C, D, _} và tần số sử dụng là A(0.35), B(0.1), C(0.2), D(0.2), _(0.15).




*Giải thuật*

```
Huffman(W[1 .. n], Σ) {
   Khởi tạo Q: Mỗi phần tử gồm (1) độ ưu tiên (tần suất xuất hiện của ký tự)
và (2) cây con một nút có nhãn là ký tự ∈ Σ;

   while (|Q| > 1) {    // Chứa nhiều hơn một phần tử
      node = extractQueue(Q);
      T_left = node.tree;
      prior_left = node.prior;

      node = extractQueue(Q);
      T_right = node.tree;
      prior_right = node.prior;

      T = createTree(T_left, T_right);
      node = makeNode(T, prior_left + prior_right);
      insertQueue(Q, node);
   }
   return   T;
}
```