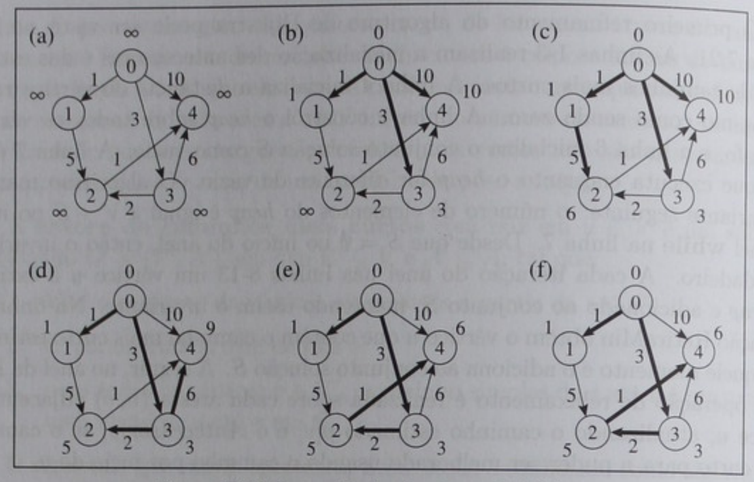

Figura 7.20 *Execução do algoritmo de Dijkstra.*

Tabela 7.1 *Valores das variáveis na execução do algoritmo de Dijkstra*

Iteração	S	p[0]	p[1]	p[2]	p[3]	p[4]
(a)	$\emptyset$	∞	∞	∞	∞	∞
(b)	$\{0\}$	0	1	∞	3	10
(c)	$\{0,1\}$	0	1	6	3	10
(d)	$\{0,1,3\}$	0	1	5	3	9
(e)	$\{0,1,3,2\}$	0	1	5	3	6
(f)	$\{0,1,3,2,4\}$	0	1	5	3	6

Assim como no algoritmo de Prim (vide Seção 7.8.2), para obter uma boa implementação para o algoritmo de Dijkstra, é preciso realizar de forma eficiente a seleção de uma nova aresta a ser adicionada à árvore formada pelas arestas em S. Durante a execução do algoritmo, todos os vértices que não estão na árvore de caminhos mais curtos residem na fila de prioridades A baseada no campo p e implementada como um *heap*, conforme mostra o Programa 7.18. Assim, para cada vértice v, $p[v]$ é o caminho mais curto obtido até o momento, de v até o vértice raiz. Como o *heap* utilizado mantém no vetor A os vértices, mas a condição do *heap* é mantida pelo caminho mais curto estimado até o momento mediante o arranjo $p[v]$, o *heap* é indireto, e o procedimento RefazInd do Programa 7.18 pode ser utilizado. Novamente, o arranjo $Pos[v]$ fornece a posição do vértice v dentro do *heap* A, permitindo assim que o vértice v possa ser acessado a um custo $O(1)$ para a operação DiminuiChaveInd.

O refinamento final do algoritmo de Dijkstra pode ser visto no Programa 7.22. O programa obtém a menor distância de um vértice origem de um grafo G a todos os outros vértices de G.

Programa 7.22 *Implementação do algoritmo de Dijkstra*

```
procedure Dijkstra (var Grafo: TipoGrafo; var Raiz: TipoValorVertice);
var Antecessor: array[TipoValorVertice] of integer;
    P         : array[TipoValorVertice] of TipoPeso;
    Itensheap : array[TipoValorVertice] of boolean;
    Pos       : array[TipoValorVertice] of TipoValorVertice;
    A         : TipoVetor;
    u, v      : TipovalorVertice;

{-- Entram aqui os operadores do tipo grafo do Programa 7.3  --}
{-- ou do Programa 7.5 ou do Programa 7.7, e os operadores   --}
{-- RefazInd, RetiraMinInd e DiminuiChaveInd do Programa 7.18--}

begin { Dijkstra }
  for u := 0 to Grafo.NumVertices do
    begin {Constroi o heap com todos os valores igual a INFINITO}
    Antecessor[u] := -1;
    p[u] := INFINITO;
    A[u+1].Chave := u; {Heap a ser construido}
    ItensHeap[u] := true;
    Pos[u] := u+1;
    end;
  n := Grafo.NumVertices;
  p[Raiz] := 0;
  Constroi (A);
  while n >= 1 do {enquanto heap nao vazio}
    begin
    u := RetiraMinInd(A).Chave;
    ItensHeap[u] := false;
    if not ListaAdjVazia (u,Grafo)
       then begin
            Aux := PrimeiroListaAdj (u,Grafo);
            FimListaAdj := false;
            while not FimListaAdj do
              begin
              ProxAdj (u, Grafo, v, Peso, Aux, FimListaAdj);
              if p[v] > p[u] + Peso
              then begin
                   p[v] := p[u] + Peso;
                   Antecessor[v] := u;
                   DiminuiChaveInd (Pos[v],p[v],A);
                   write ('Caminho: v[',v,'] v[',Antecessor[v],']',
                          ' d[',p[v],']');readln;
                   end;
              end;
            end;
    end;
end; { Dijkstra }
```