**Instituto**

**Politécnico**

**Nacional**

**Escuela Superior de Cómputo**

**REPORTE DE LA SESIÓN 8: Grafos**

**Materia:**

Análisis de algoritmos

**Grupo:**

3CM3

**Integrantes:**

Castro Cruces Jorge Eduardo

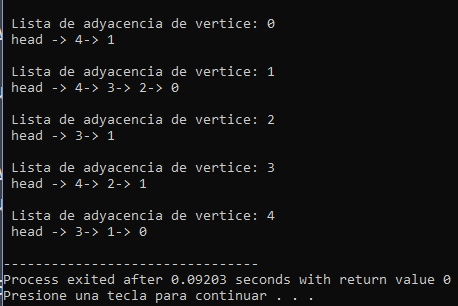
**Fecha:**

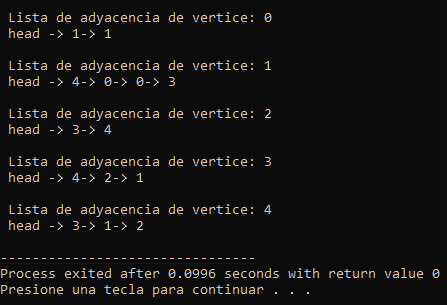
Lunes, junio 15, 2020

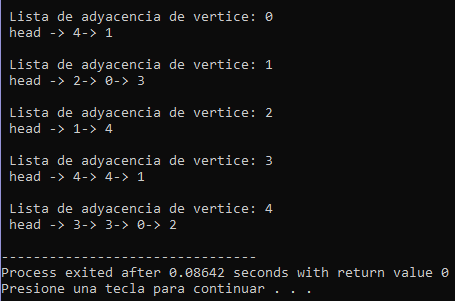
**Código**

1. #include <stdio.h>
2. #include <stdlib.h>
4. **struct** AdjListNode{
5. **int** dest;
6. **struct** AdjListNode\* next;
7. };
9. **struct** AdjList{
10. **struct** AdjListNode \*head;
11. };
13. **struct** Graph{
14. **int** V;
15. **struct** AdjList\* array;
16. };
18. **struct** AdjListNode\* newAdjListNode(**int** dest){
19. **struct** AdjListNode\* newNode = (**struct** AdjListNode\*) malloc(**sizeof**(**struct** AdjListNode));
20. newNode->dest = dest;
21. newNode->next = NULL;
22. **return** newNode;
23. }
25. **struct** Graph\* createGraph(**int** V){
26. **struct** Graph\* graph = (**struct** Graph\*) malloc(**sizeof**(**struct** Graph));
27. graph->V = V;
28. graph->array = (**struct** AdjList\*) malloc(V \* **sizeof**(**struct** AdjList));
29. **int** i;
30. **for** (i = 0; i < V; ++i)
31. graph->array[i].head = NULL;
32. **return** graph;
33. }
35. **void** addEdge(**struct** Graph\* graph, **int** src, **int** dest){
36. **struct** AdjListNode\* newNode = newAdjListNode(dest);
37. newNode->next = graph->array[src].head;
38. graph->array[src].head = newNode;
39. newNode = newAdjListNode(src);
40. newNode->next = graph->array[dest].head;
41. graph->array[dest].head = newNode;
42. }
44. **void** printGraph(**struct** Graph\* graph){
45. **int** v;
46. **for** (v = 0; v < graph->V; ++v){
47. **struct** AdjListNode\* pCrawl = graph->array[v].head;
48. printf("\n Lista de adyacencia de vertice: %d\n head ", v);
49. **while** (pCrawl){
50. printf("-> %d", pCrawl->dest);
51. pCrawl = pCrawl->next;
52. }
53. printf("\n");
54. }
55. }
57. **int** main(){
58. **int** V = 5;
59. **struct** Graph\* graph = createGraph(V);
60. addEdge(graph, 0, 1);
61. addEdge(graph, 0, 4);
62. addEdge(graph, 1, 2);
63. addEdge(graph, 1, 3);
64. addEdge(graph, 1, 4);
65. addEdge(graph, 2, 3);
66. addEdge(graph, 3, 4);
67. printGraph(graph);
68. **return** 0;
69. }

**Capturas de pantalla**







**Complejidad**

**Pros:** Ahorra espacio O (| V | + | E |). En el peor de los casos, puede haber un número C (V, 2) de aristas en un gráfico que consume espacio O (V ^ 2). Agregar un vértice es más fácil.

**Contras:** las consultas como si hay un borde desde el vértice u al vértice v no son eficientes y se pueden hacer O (V).