Checklist Source Code Graf dalam Bahasa C dan C++

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Algoritma** | **C** | **C++** |
| 1 | Breadth First Search (BFS) | ✓ |  |
| 2 | Depth First Search (DFS) | ✓ |  |
| 3 | Prim | ✓ |  |
| 4 | Djikstra | ✓ |  |
| 5 | Bellman-Ford | ✓ |  |
| 6 | Floyd-Warshall | ✓ |  |
| 7 | Kruskal | ✓ |  |

Checklist Source Code Pohon dalam Bahasa C dan C++

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Algoritma** | **C** | **C++** |
| 1 | Binary Search Tree (BST) | ✓ |  |
| 2 | Digital Search Tree |  |  |
| 3 | Radix Tree | ✓ | ✓ |
| 4 | Adelson-Velskii and Landis (AVL) | ✓ | ✓ |
| 5 | Red-Black Tree | ✓ |  |
| 6 | Search Tree Traversal |  |  |
| 7 | Splay Tree |  | ✓ |

|  |
| --- |
| **Graf - Breadth First Search (BFS)** |
| **#include <stdlib.h>**  int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;  **void bfs(int v)**{  for (i = 1; i <= n; i++){  if (a[v][i] && !visited[i])  q[++r] = i;  }  if (f <= r){  visited[q[f]] = 1;  bfs(q[f++]);  }  }  **int main()**{  int v;  printf("\n Number of vertices: 7");  n = 7;  for (i = 1; i <= n; i++){  q[i] = 0;  visited[i] = 0;  }  printf("\n Enter graph data in matrix form:\n");  for (i = 1; i <= n; i++)  for (j = 1; j <= n; j++)  scanf("%d", &a[i][j]);    printf("\n Enter the starting vertex:");  scanf("%d", &v);  bfs(v);    printf("\n The node which are reachable are:\n");  for (i = 1; i <= n; i++)  if (visited[i])  printf("%d\t", i);  else  printf("\n Bfs is not possible");    return 0;  } |

|  |
| --- |
| **Execution Trace Format JSON** (Multidimensi Array, Array, Primitif Variabel Data) |
| {  **"code"**: "#include <stdlib.h>\n\nint a[2][2],\n q[4],\n visited[4],\n ... }",  **"trace"**: **[{**  **"event"**: "step\_line",  **"func\_name"**: "main",  **"globals"**: {  "a": ["C\_MULTIDIMENSIONAL\_ARRAY", "0x6010E0", [2, 2],  ["C\_DATA", "0x6010E0", "int", 0],  ["C\_DATA", "0x6010E4", "int", 0],  ["C\_DATA", "0x6010E8", "int", 0],  ["C\_DATA", "0x6010EC", "int", 0]  ],  "f": ["C\_DATA", "0x601064", "int", 0],  "i": ["C\_DATA", "0x601270", "int", 0],  "j": ["C\_DATA", "0x601084", "int", 0],  "n": ["C\_DATA", "0x601080", "int", 0],  "q": ["C\_ARRAY", "0x601280", ["C\_DATA", "0x601280", "int", 0],  ["C\_DATA", "0x601284", "int", 0],  ["C\_DATA", "0x601288", "int", 0],  ["C\_DATA", "0x6012A4", "int", 0]  ],  "r": ["C\_DATA", "0x601050", "int", -1],  "visited": ["C\_ARRAY", "0x6010A0", ["C\_DATA", "0x6010A0", "int", 0],  ["C\_DATA", "0x6010A4", "int", 0],  ["C\_DATA", "0x6010A8", "int", 0],  ["C\_DATA", "0x6010C4", "int", 0]  ]  },  **"heap"**: {},  **"line"**: 28,  **"ordered\_globals"**: ["a", "q", "visited", "n", "i", "j", "f", "r"],  **"stack\_to\_render"**: [{  "encoded\_locals": {  "v": ["C\_DATA", "0xFFF000BDC", "int", "<UNINITIALIZED>"]  },  "frame\_id": "0xFFF000BF0",  "func\_name": "main",  "is\_highlighted": true,  "is\_parent": false,  "is\_zombie": false,  "line": 28,  "ordered\_varnames": ["v"],  "parent\_frame\_id\_list": [],  "unique\_hash": "main\_0xFFF000BF0"  }],  **"stdout"**: ""  }, { . . . **}]**  } |