7.9

- 1->5->2->3->6->4
- 1->5->2->6->3->4
- 1->5->6->2->3->4
- 5->1->2->3->6->4
- 5->1->2->6->3->4
- 5->1->6->2->3->4
- 5->6->1->2->3->4
- 其中，用7.5.1节算法求得为5->6->1->2->3->4

7.10

关键路径为a->G->H->K->J->E->w

	a	A	B	C	D	E	F	G	H	I	J	K	w
e	0	1	6	17	3	34	4	3	13	1	31	22	44
l	0	20	24	26	19	34	8	3	13	7	31	22	44

	<a, A>	<a, B>	<a, D>	<a, F>	<a, G>	<a, I>	<A, C>	<B, C>
e	0	0	0	0	0	0	1	6
l	19	18	16	4	0	6	20	24
	<D, C>	<D, E>	<D, J>	<F, E>	<F, H>	<G, w>	<G, H>	<I, H>
e	3	3	3	4	4	3	3	1
l	19	26	25	23	8	23	3	7
	<C, E>	<H, C>	<H, J>	<H, K>	<K, J>	<J, E>	<J, w>	<E, w>
e	17	13	13	13	22	31	31	34
l	26	22	27	13	22	31	32	34

7.11

	a	b	c	d	e	f	g	S
0	0	15	2	12	INF	INF	INF	{a}
1	0	15	2	12	10	6	INF	{a, c}
2	0	15	2	11	10	6	16	{a, c, f}
3	0	15	2	11	10	6	16	{a, c, e, f}
4	0	15	2	11	10	6	14	{a, c, e, f, d}
5	0	15	2	11	10	6	14	{a, c, e, f, d, g}
6	0	15	2	11	10	6	14	{a, c, e, f, d, g, b}

7.14

```
struct Arc {
    Arc* nxt;
    int node;
}
struct Node{
    int val;
    Arc* nxt;
}
unordered_map<int, Node> G;
void addEdge(int begin, int end) {
    Arc *newArc = new Arc();
    newArc->nxt = G[begin].nxt;
    G[begin].nxt = newArc;
    newArc->node = end;
}
void Init() {
    int n, m;
    cin >> n >> m;
    for (int i = 0; i < n; i++) {
        int val;
        cin >> val;
        G[val].val = val;
        G[val].nxt = nullptr;
    }
    for (int i = 0; i < m; i++) {
        int begin, end;
        cin >> begin >> end;
        addEdge(begin, end);
    }
}
```

7.15

```
void InsertVex(G, v) {
    G.vexMap[v] = true;
}
void DeleteVex(G, v) {
    G.vexMap.erase(v);
}
void InsertArc(G, v, w) {
    G.arcMap[v][w] = true;
}
void DeleteArc(G, v, w) {
    G.arcMap[v][w] = false;
}
```

7.19

```
struct Node{
    int val;
    Arc *nxt;
}
struct Arc{
    int lNode, rNode;
    Arc *lnxt, *rnxt;
}
void addEdge(int begin, int end) {
    Arc *newArc = new Arc();
    newArc->lNode = begin;
    newArc->rNode = end;
    newArc->lnxt = G[begin].nxt;
    G[begin].nxt = newArc;
    newArc->rnxt = G[end].nxt;
    G[end].nxt = newArc;
}
vector<Node> G;
void Init() {
    int n, m;
    cin >> n >> m;
    for (int i = 0; i < n; i++) {
        int val;
        cin >> val;
        G.push_back({val, nullptr});
    }
    for (int i = 0; i < m; i++) {
        int begin, end;
        cin >> begin >> end;
        addEdge(begin, end);
    }
}
```

7.22

```cpp
vector<vector<int>> G;
vector<bool> vis;
bool dfs(int cur, int tar) {
    if (cur == tar) return true;
    vis[cur] = true;
    bool ans = false;
    for (auto &x : G[cur])
        if (!vis[x])
            ans |= dfs(x, tar);
    return ans;
}
bool solve() {
    int n, m;
    cin >> n >> m;
    G.resize(n);
    vis.resize(n);
    for (int i = 0; i < m; i++) {
        int x, y;
        cin >> x >> y;
        G[x],push_back(y);
    }
    int vi, vj;
    cin >> vi >> vj;
    return dfs(vi, vj);
}
```

7.23

```cpp
vector<vector<int>> G;
vector<bool> vis;
bool bfs(int begin, int tar) {
    queue<int> my_queue;
    my_queue.push(begin);
    vis[begin] = true;
    while (!my_queue.empty()) {
        int now = my_queue.front();
        my_queue.pop();
        for (auto &x : G[now])
            if (!vis[x]) {
                my_queue.push(x);
                vis[x] = true;
                if (x == tar) return true;
            }
    }
    return false;
}
bool solve() {
    int n, m;
    cin >> n >> m;
    G.resize(n);
    vis.resize(n);
    for (int i = 0; i < m; i++) {
        int x, y;
        cin >> x >> y;
```

```
        G[x],push_back(y);
    }
    int vi, vj;
    cin >> vi >> vj;
    return bfs(vi, vj);
}
```

7.24

```
vector<vector<int>> G;
vector<bool> vis;
void solve() {
    int n, m;
    cin >> n >> m;
    G.resize(n);
    vis.resize(n);
    for (int i = 0; i < m; i++) {
        int x, y;
        cin >> x >> y;
        G[x],push_back(y);
    }
    int vi, vj;
    cin >> vi >> vj;
    stack<int> my_stack;
    my_stack.push(1);
    while (!my_stack.empty()){
        int cur = my_stack.top();
        my_stack.pop();
        vis[cur] = true;
        for (auto &x : G[cur])
            if (!vis[cur]) my_stack.push(x);
    }
}
```

7.34

```
void solve() {
    int tot = 0;
    queue<int> my_queue;
    for (int i = 0; i < n; i++)
        if (!in[i]) my_queue.push(i);
    while (!my_queue.empty()) {
        int cur = my_queue.front();
        my_queue.pop();
        idx[cur] = tot++;
        for (auto &x : G[cur]) {
            in[x]--;
            if (!in[x]) my_queue.push(x);
        }
    }
}
```

7.35

```cpp
int tot = 0;
vector<bool> vis;
void dfs(int cur) {
    tot++;
    vis[cur] = true;
    for (auto &x : G[cur])
        if (!vis[cur]) dfs(x);
}
int solve() {
    int id = -1;
    for (int i = 0; i < n; i++)
        if (!in[i]) {
            if (id == -1) id = i;
            else return -1      // -1 for no
        }
    vis.resize(n);
    dfs(id);
    return tot == n ? id : -1;
}
```

7.38

```cpp
void dfs(int cur) {
    if (G[cur].size()) {
        dfs(G[cur][0]);
        dfs(G[cur][1]);
    }
    print(data[cur]);
}
void solve() {
    for (int i = 0; i < n; i++)
        if (!in[i]) dfs(i)
}
```