题目描述

求 单向链表 中间的节点值，如果奇数个节点取中间，偶数个取偏右边的那个值。

输入描述

第一行 链表头节点地址 后续输入的节点数n

后续输入每行表示一个节点，格式 节点地址 节点值 下一个节点地址(-1表示 空指针)

输入保证链表不会出现环，并且可能存在一些节点不属于链表。

输出描述

单向链表中间的节点值

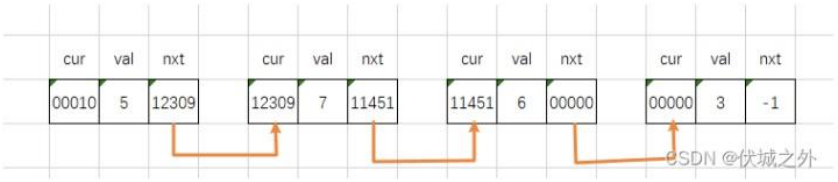
用例

输入	00010 4 00000 3 -1 00010 5 12309 11451 6 00000 12309 7 11451
输出	6
说明	无

输入	10000 3 76892 7 12309 12309 5 -1 10000 1 76892
输出	7
说明	无

题目解析

用例1示意图如下



JS本题可以利用数组模拟链表

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
let head;
let n;
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 1) {
    [head, n] = lines[0].split(" ");
  }

  if (n && lines.length === n - 0 + 1) {
    lines.shift();
```

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
let head;
let n;
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 1) {
    [head, n] = lines[0].split(" ");
  }

  if (n && lines.length === n - 0 + 1) {
    lines.shift();

    const nodes = {};

    lines.forEach((line) => {
      const [addr, val, nextAddr] = line.split(" ");
      nodes[addr] = [val, nextAddr];
    });

    console.log(getResult(head, nodes));

    lines.length = 0;
  }
});

function getResult(head, nodes) {
  const linkedlist = [];

  let node = nodes[head];
  while (node) {
    const [val, next] = node;

    linkedlist.push(val);
    node = nodes[next];
  }

  const len = linkedlist.length;

  const mid = len % 2 === 0 ? len / 2 : Math.floor(len / 2);

  return linkedlist[mid];
}
```

Java算法源码

```java
import java.util.HashMap;
import java.util.LinkedList;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String head = sc.next();
    int n = sc.nextInt();

    HashMap<String, String[]> nodes = new HashMap<>();
    for (int i = 0; i < n; i++) {
      String addr = sc.next();
      String val = sc.next();
      String nextAddr = sc.next();
      nodes.put(addr, new String[] {val, nextAddr});
    }

    System.out.println(getResult(head, nodes));
  }

  public static String getResult(String head, HashMap<String, String[]> nodes) {
    LinkedList<String> link = new LinkedList<>();

    String[] node = nodes.get(head);
    while (node != null) {
      String val = node[0];
      String next = node[1];

      link.add(val);
      node = nodes.get(next);
    }

    int len = link.size();
    int mid = len / 2;
```

Java算法源码

```java
import java.util.HashMap;
import java.util.LinkedList;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String head = sc.next();
    int n = sc.nextInt();

    HashMap<String, String[]> nodes = new HashMap<>();
    for (int i = 0; i < n; i++) {
      String addr = sc.next();
      String val = sc.next();
      String nextAddr = sc.next();
      nodes.put(addr, new String[] {val, nextAddr});
    }

    System.out.println(getResult(head, nodes));
  }

  public static String getResult(String head, HashMap<String, String[]> nodes) {
    LinkedList<String> link = new LinkedList<>();

    String[] node = nodes.get(head);
    while (node != null) {
      String val = node[0];
      String next = node[1];

      link.add(val);
      node = nodes.get(next);
    }

    int len = link.size();
    int mid = len / 2;
    return link.get(mid);
  }
}
```

Python算法源码

复制

```python
# 输入获取
head, n = input().split()

nodes = {}
for i in range(int(n)):
    addr, val, nextAddr = input().split()
    nodes[addr] = [val, nextAddr]


# 算法入口
def getResult(head, nodes):
    linkedlist = []
    node = nodes.get(head)

    while node is not None:
        val, next = node
        linkedlist.append(val)
        node = nodes.get(next)

    length = len(linkedlist)
    mid = int(length / 2)

    return linkedlist[mid]


# 算法调用
print(getResult(head, nodes))
```