# 题目

给定一个单链表，随机选择链表的一个节点，并返回相应的节点值。保证每个节点被选的概率一样。

进阶:

如果链表十分大且长度未知，如何解决这个问题？你能否使用常数级空间复杂度实现？

示例：

// 初始化一个单链表 [1,2,3].

ListNode head = new ListNode(1);

head.next = new ListNode(2);

head.next.next = new ListNode(3);

Solution solution = new Solution(head);

// getRandom()方法应随机返回1,2,3中的一个，保证每个元素被返回的概率相等。

solution.getRandom();

# 分析

**思路：**

蓄水池算法，核心就是：

对于1概率是1，

对于2概率是1/2，

对于3就是概率1/3，

一次类推直到1/n为止。

**代码：**

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode() : val(0), next(nullptr) {}

\* ListNode(int x) : val(x), next(nullptr) {}

\* ListNode(int x, ListNode \*next) : val(x), next(next) {}

\* };

\*/

class Solution {

private:

ListNode\* head;

public:

/\*\* @param head The linked list's head.

Note that the head is guaranteed to be not null, so it contains at least one node. \*/

Solution(ListNode\* head) : head(head) {

}

/\*\* Returns a random node's value. \*/

int getRandom() {

int k = 2;

int res = head->val;

ListNode\* curr = head->next;

while (curr != nullptr)

{

if (rand() % k == 0)

{

res = curr->val;

}

curr = curr->next;

++k;

}

return res;

}

};

/\*\*

\* Your Solution object will be instantiated and called as such:

\* Solution\* obj = new Solution(head);

\* int param\_1 = obj->getRandom();

\*/