动态规划

资源

1. 最长递增子序列
2. LeetCode300

开始学习

Vue源码调试

1. 下载vue3：`git clone https://github.com/vuejs/vue-next.git`
2. 在vue-next下安装依赖：`yarn --ignore-scripts`
3. 生成sourcemap文件，package.json

   `"dev": "node scripts/dev.js --sourcemap"`
4. 编译: `yarn dev`

   生成结果:

   packages\vue\dist\vue.global.js

   packages\vue\dist\vue.global.js.map
5. 调试范例代码： `yarn serve`
6. 在vue/examples下创建patch.html

```html
<div id="app">
    <button v-on:click="add">{{count}}</button>

    <ul>
        <li v-for="item in state.arr" :key="item">{{item}}</li>
    </ul>

</div>

<script src="../dist/vue.global.js"></script>
<!-- <script src="http://unpkg.com/vue@next"></script> -->


<script>
    const {
            createApp,
            ref,
            reactive
        } = Vue
        const app = createApp({
            setup() {
                const count = ref(0)
                const state = reactive({
                    arr: [0, 1, 2, 3, 4]
                })
                const add = () => {
                    count.value++
                    if (count.value % 2) {
                        state.arr = [1, 3, 2, 5]
                    } else {
                        state.arr = [0, 1, 2, 3, 4]

                    }
                }

                return {
                    count,
                    add,
                    state
                }
            }
        }).mount('#app')
    </script>
```

7. 打开地址：http://localhost:5000/packages/vue/examples/patch

patch VS diff

最长递增子序列

场景：Vue中更新阶段，新老虚拟dom的diff的时候，如果有节点移动，那么此时可以计算下dom节点中最长递增子序列，减少move，确保对dom的操作影响到最小。

```
function getSequence(arr: number[]): number[] {
  const p = arr.slice()
  const result = [0]
  let i, j, u, v, c
  const len = arr.length
  for (i = 0; i < len; i++) {
    const arrI = arr[i]
    if (arrI !== 0) {
      j = result[result.length - 1]
      if (arr[j] < arrI) {
        p[i] = j
        result.push(i)
        continue
      }
      u = 0
      v = result.length - 1
```

```
        while (u < v) {
          c = (u + v) >> 1
          if (arr[result[c]] < arrI) {
            u = c + 1
          } else {
            v = c
          }
        }
        if (arrI < arr[result[u]]) {
          if (u > 0) {
            p[i] = result[u - 1]
          }
          result[u] = i
        }
      }
    }
    u = result.length
    v = result[u - 1]
    while (u-- > 0) {
      result[u] = v
      v = p[v]
    }
    return result
  }
```