lab5 report

algorithm

- Use dfs. The function dfs() takes one integer i as the argument, and return a boolean value, representing if the current chosen chapters cover 1-N.
- The function matains an array `vis[]` to record which numbers have been covered. If `vis[j] = 0` for a number `j`, `j` has not been covered, so it can be added to the chosen chapters.
- In `dfs(i)`, if the first number `n` on the `(i+1)th` page can be chosen, then let `vis[n] = 1` and then call `dfs(i+1)` recursively. If the return value of `dfs(i+1)` is true, then `dfs(i)` returns true. If the return value is false, let `vis[n] = 0` (backtrack) and do the same to the other number on the `(i+1)th` page. If both of them cannot be chosen, return false. The recursion ends if `i == N` and returns true.

essential parts of code

```
AND R1, R1, #0
ADD R6, R6, #-1
STR R1, R6, #0 ; push the argument onto user stack
JSR DFS ; call DFS(0)

DFS     ADD R6, R6, #-1
        STR R7, R6, #0
        AND R0, R0, #0
        ADD R0, R0, #1 ; when DFS is called, push R7 onto user stack, let R0(return value) be 1
        LD R3, save_N
        LEA R3, ARRAY
        ADD R3, R3, #2
        ADD R3, R3, R1
        ADD R3, R3, R1
        LDR R2, R3, #0
        LEA R4, vis
        ADD R4, R4, R2
        LDR R4, R4, #0 ; vis[R2] = 0
        NOT R3, R3
        ADD R3, R3, #1
        ADD R3, R1, R3
        BRz RETURN ; if(R1 == N) return true

CHECK_1 ; load the first number on the (i+1)th card to R2
        LEA R3, ARRAY
        ADD R3, R3, #2
        ADD R3, R3, R1
        ADD R3, R3, R1
        LDR R2, R3, #0
        LEA R4, vis
        ADD R4, R4, R2
        LDR R4, R4, #0 ; R4 = vis[R2]
```

```
        BRp CHECK_2 ; if R4 is 1, it cannot be chosen, check the other number
        ; vis[R2] <- 1, detail omitted
        ADD R1, R1, #1 ; i + 1
        ADD R6, R6, #-1
        STR R1, R6, #0 ; push (i+1) onto user stack
B       JSR DFS ; call DFS(i+1) recursively
        ADD R0, R0, #0 ; when PC returns to here, the return value of DFS(i+1) is R0
        BRz BK_1 ; if R0 is 0, backtrack
        ; the first number on the (i+1)th card should be chosen, let res[i+1] = R2, detail omitted
        ;R2 = ARRAY[2 * i]
        LEA R3, ARRAY
        ADD R3, R3, R1
        ADD R3, R3, R1
        LDR R2, R3, #0

        LEA R4, vis
        ADD R4, R4, R1
        STR R2, R4, #0 ; res[R1] = R2
        AND R0, R0, #0
        ADD R0, R0, #1
        BRnzp RETURN ; return true

BK_1    ;vis[R2] <- 0, detail omitted

CHECK_2 ; load the second number on the (i+1)th card to R2, detail omitted
        LDR R4, R4, #0 ; R4 = vis[R2]
        BRp CHECK_3 ; if R4 is 1, it cannot be chosen, return false
        ; vis[R2] <- 1, detail omitted
        ADD R1, R1, #1 ; i + 1
        ADD R6, R6, #-1
        STR R1, R6, #0 ; push (i+1) onto user stack
C       JSR DFS ; call DFS(i+1) recursively
        ADD R0, R0, #0 ; when PC returns to here, the return value of DFS(i+1) is R0
        BRz BK_1 ; if R0 is 0, backtrack
        ; the second number on the (i+1)th card should be chosen, let res[i+1] = R2, detail omitted

        BRnzp RETURN ; return true

BK_2    ;vis[R2] <- 0, detail omitted


CHECK_3 AND R0, R0, #0 ; return false, clear R0

RETURN  LDR R7, R6, #0
        LDR R1, R6, #1
        ADD R6, R6, #2 ; Pop R1 and R7 from user stack and return R0
        RET
```

use of some registers and memory

- R0: the return value.
- R1: the argument of `DFS()`
- R7: return address
- `ARRAY .BLKW #40` : array to store numbers, the first number on the `i`th card is at `ARRAY[2*i+1]`.
- `vis .BLKW #20`: array to check if a number has been chosen to the current number set.
- `res .BLKW #20`: array to store the result, `res[i]` is the number to be chosen on the `i`th card.

problems

> When `DFS` is called recursively, what happens to the user stack?

The push and pop operations happen at these positions:

1. When `DFS(i)` is to be called, push the argument `i` onto user stack.
2. At the beginning of `DFS(i)`, push R7 onto user stack.
3. When `DFS(i+1)` returns, pop R1 and R7 for the processing of `DFS(i)`.

Before `DFS(0)` is called, R1 is 0 and R7 is address of the subroutine to output result, push them onto user stack. Then check numbers on the first card. If the first can be chosen, call the `DFS` at label `B`, and then `1` and `B+1` are pushed. Keep pushing, until the argument equals `N` (return true) or no number can be chosen(return false). Before return, pop R1 and R7 from user stack and jump to R7, continue to push(if there are number to chosen) or pop(no number can be chosen), and backtrack or update the result corresponding to the R0.