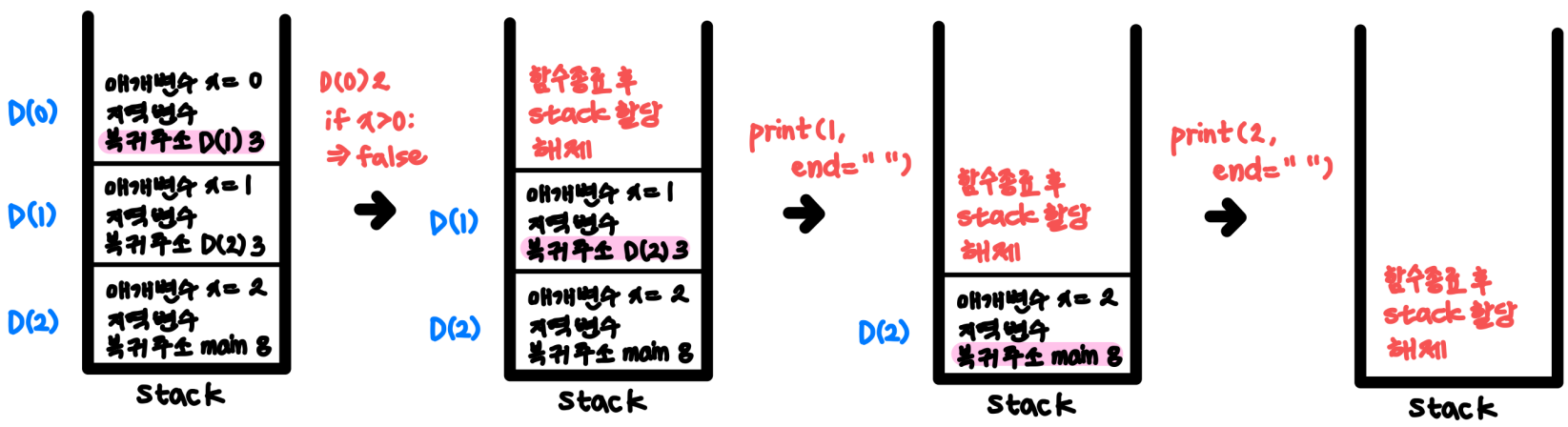
6. 완전탐색 (백트랙킹, 상태트리와 CUT EDGE)-DFS(깊이우선탐색)기초

Created	@September 17, 2022 1:45 PM
Progress	In Progress

6.0. [선수지식] 재귀함수와 스택(중요)

```
def DFS(x):
  if x > 0:
    DFS(x-1)
    print(x, end=" ")

if __name__=="__main__":
    n = 2
    DFS(n)
```



stack 할당

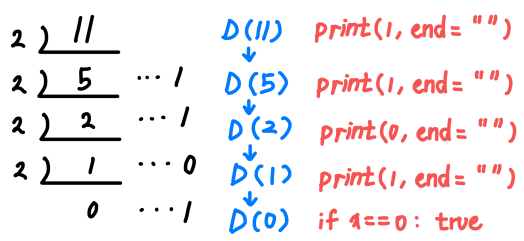
> 출력 결과: ***1 2***

6.1. 재귀함수를 이용한 이진수 출력

```
def DFS(x):
  if x == 0:
    return
  else:
    DFS(x // 2)    # 6.1.2. 올바르게 출력되는 경우
    print(x % 2, end="")
#   DFS(x // 2)    # 6.1.1. 거꾸로 출력되는 경우
```

```
if __name__=="__main__":
  n = 11
  DFS(n)
```

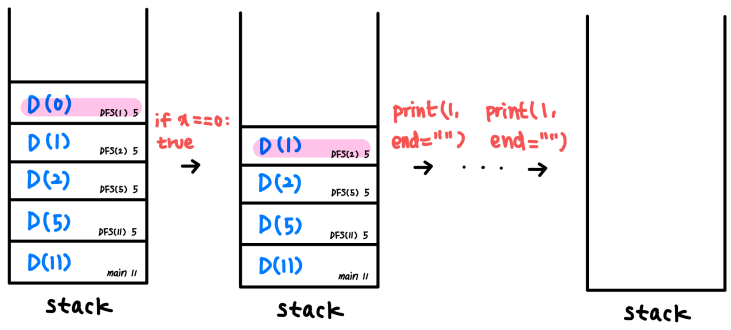
6.1.1. 거꾸로 출력되는 경우



연산 예(좌)거꾸로 출력되는 경우 상태 트리 설계(우)

> 출력 결과: *1101*

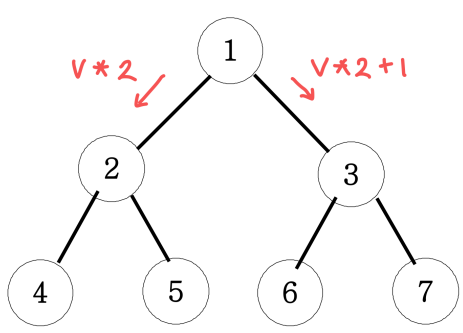
6.1.2. 올바르게 출력되는 경우



올바른 stack 할당

> 출력 결과: *1011*

6.2. 이진트리순회(DFS: Depth First Search)



이진트리

6.2.1. 전위순회

> 순회 순서: *부모-왼쪽 자식-오른쪽 자식*

```
def DFS(v):
  if v > 7:
    return
  else:
    print(v, end=" ")
    DFS(v * 2)
    DFS(v * 2 + 1)

if __name__=="__main__":
  DFS(1)
```

> 출력 결과: *1 2 4 5 3 6 7*

6.2.2. 중위순회

> 순회 순서: *왼쪽 자식-부모-오른쪽 자식*

```
def DFS(v):
  if v > 7:
    return
  else:
    DFS(v * 2)
    print(v, end=" ")
    DFS(v * 2 + 1)

if __name__=="__main__":
  DFS(1)
```

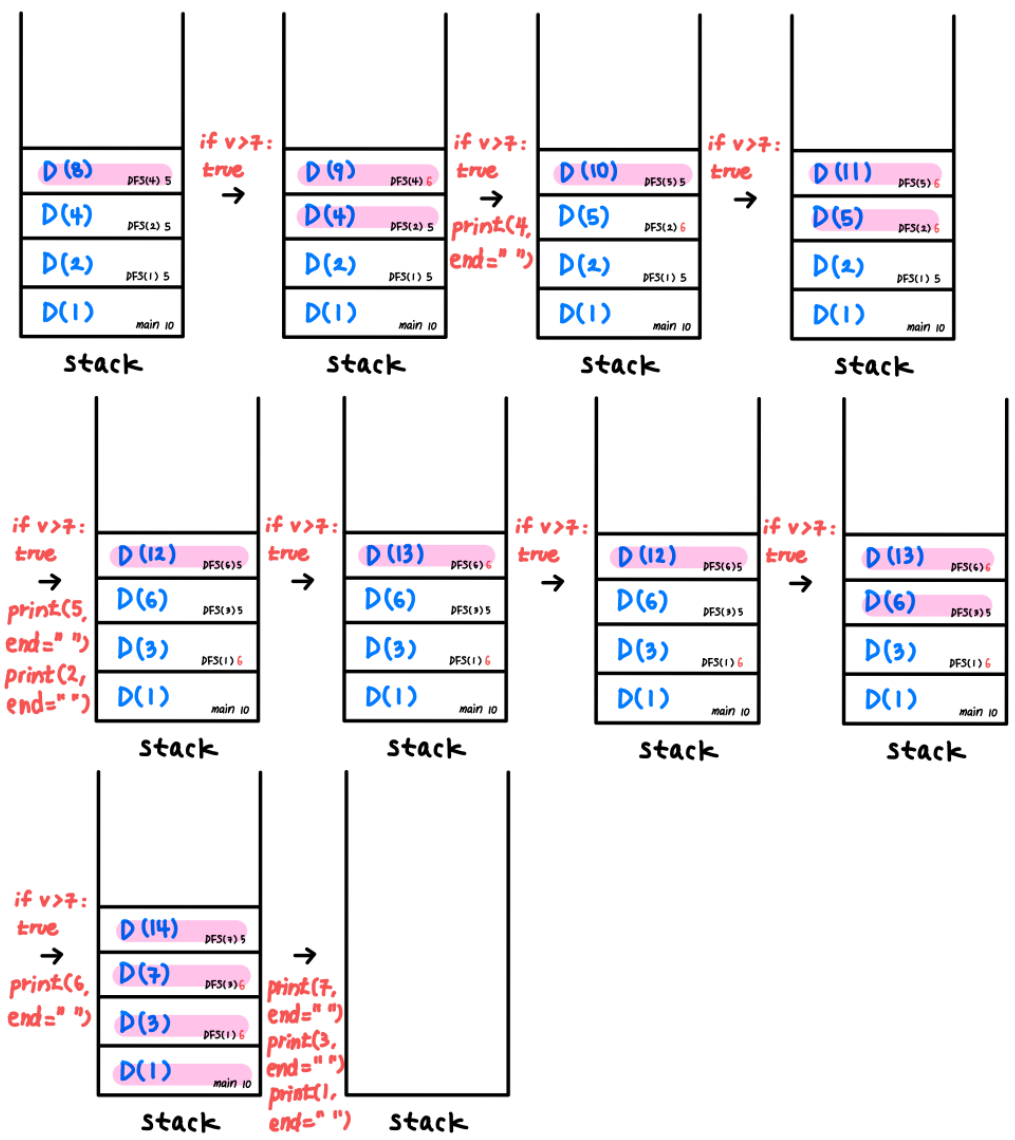
> 출력 결과: *4 2 5 1 6 3 7*

6.2.3. 후위순회

> 순회 순서: *왼쪽 자식-오른쪽 자식-부모*

```
def DFS(v):
  if v > 7:
    return
  else:
    DFS(v * 2)
    DFS(v * 2 + 1)
    print(v, end=" ")

if __name__=="__main__":
  DFS(1)
```

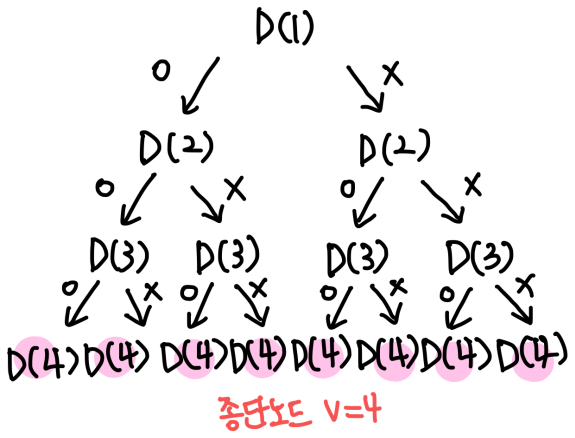


후위순회 stack 할당

> 출력 결과: *4 5 2 6 7 3 1*

- 병합정렬

6.3. 부분집합 구하기(DFS)



상태트리 설계

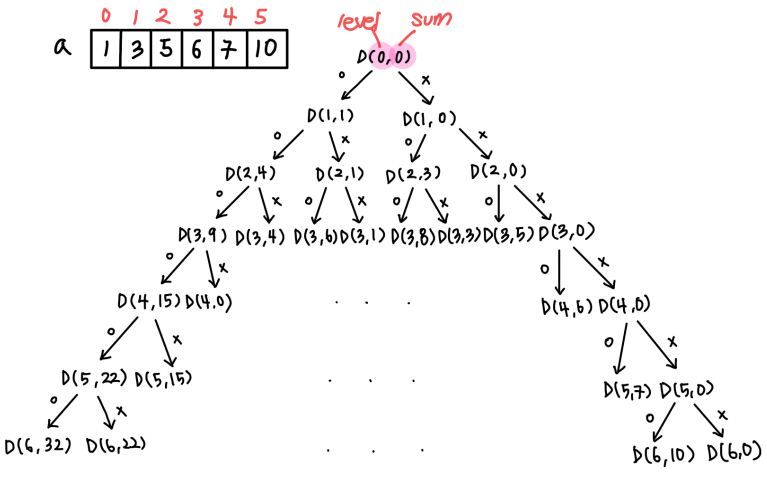
```
def DFS(v):
  if v == n + 1:
    for i in range(1, n + 1):
      if visited[i] == 1:
        print(i, end=" ")
    print()
  else:
    visited[v] = 1
    DFS(v + 1)
    visited[v] = 0
    DFS(v + 1)

if __name__=="__main__":
  n = 3
  visited = [0] * (n + 1)
  DFS(1)
```

6.4. 합이 같은 부분집합(DFS: 아마존 인터뷰)

- 두 개의 부분집합으로 나누었을 때 두 부분집합의 원소의 합이 서로 같은 경우

> *sum == total - sum*



상태트리 설계

```
import sys


def DFS(L, sum):
  if L == n:
    if sum == total - sum:
      print("YES")
      sys.exit(0)
  else:
    DFS(L + 1, sum + a[L])
    DFS(L + 2, sum)

if __name__=="__main__":
  n = 6
  a = [1, 3, 5, 6, 7, 10]
  total = sum(a)
  DFS(0, 0)
  print("NO")
```

- `sys.exit()`: `sys` 모듈의 프로그램 종료 함수

- `sys.exit(0)` : *Process finished with exit code 0*
- `sys.exit(1)` : *Process finished with exit code 1*

6.4.1. 시간복잡도 단축

> *sum이 total//2를 넘어가면 중복*

```
import sys


def DFS(L, sum):
  if sum > total // 2:
    return
  if L == n:
    if sum == total - sum:
      print("YES")
      sys.exit(0)
  else:
    DFS(L + 1, sum + a[L])
    DFS(L + 2, sum)

if __name__=="__main__":
  n = 6
  a = [1, 3, 5, 6, 7, 10]
  total = sum(a)
  DFS(0, 0)
  print("NO")
```

- `//`: 나누기 소수점 아래 결과 버림

6.5. 바둑이 승차-Cut Edge Tech

```
import sys


def DFS(L, sum, tsum):
  global result
  if sum + (total-tsum) < result:
    return
  if sum > c:
    return
  if sum == c:
    print(sum)
    sys.exit(0)
  if L == n:
    result = max(result, sum)
    return
  else:
    DFS(L+1, sum + weights[L], tsum + weights[L])
    DFS(L+1, sum, tsum + weights[L])

if __name__=="__main__":
  c = 259
  n = 5
  result = -2147000000
  weights = [81, 58, 42, 33, 61]
  total = sum(weights)
  DFS(0, 0, 0)
  print(result)
```

- `-2147000`: 가장 작은 수

> 시간 초과 해결: *남은 노드들을 미리 검사함*

6.6. 중복순열 구하기(DFS)

```
def DFS(L):
  global cnt
  if L == m:
    for r in res:
      print(r, end=" ")
    print()
    cnt += 1
```

```
  else:
    for i in range(1, n + 1):
      res[L] = i
      DFS(L + 1)

if __name__=="__main__":
  n = 3
  m = 2
  res = [0] * m
  cnt = 0
  DFS(0)
  print(cnt)
```

6.7. 동전 교환-Cut Edge Tech

```
def DFS(L, sum):
  global result
  if L > result:
    return
  if sum > m:
    return
  if sum == m:
    result = min(result, L)
  else:
    for i in range(n):
      DFS(L + 1, sum + coins[i])

if __name__=="__main__":
  n = 3
  coins = [1, 2, 5]
  coins.sort(reverse=True)
  m = 15
  result = 2147000000
  DFS(0, 0)
  print(result)
```

- `2147000`: 가장 큰 수

> `coins.sort(reverse=True)`: *가장 적게 뽑기 위해서는 큰 수부터 확인해야 함*

> 시간 초과 해결: *현재 개수와 현재 상태의 최소 개수를 미리 검사함*

6.8. 순열 구하기(DFS)

```
def DFS(L):
  global cnt
  if L == m:
    for r in res:
      print(r, end=" ")
    print()
    cnt += 1
    return
  else:
    for i in range(1, n + 1):
      if visited[i] == 0:
        visited[i] = 1
        res[L] = i
        DFS(L + 1)
        visited[i] = 0

if __name__=="__main__":
  n = 3
  m = 2
  visited = [0] * (n + 1)
  res = [0] * m
  cnt = 0
  DFS(0)
  print(cnt)
```

6.9. 순열 추측하기(순열, 파스칼 응용)

```
def DFS(L):
  if L == n:
```

```
    sum = pascal[0][0] + pascal[0][n - 1]
    tsum = 0
    for i in range(1, n - 1):
      tsum += pascal[0][i]
    sum += tsum * 3
    if sum == f:
      result = min(result,
  else:
    for i in range(1, n + 1):
      if visited[i] == 0:
        visited[i] = 1
        pascal[0][L] = i
        DFS(L + 1)
        visited[i] = 0

if __name__=="__main__":
  n = 4
  f = 16
  visited = [0] * (n + 1)
  pascal = [[0] * n] * n
  result = "9999"
  DFS(0)
  result.sort()
  print(result[0])
```