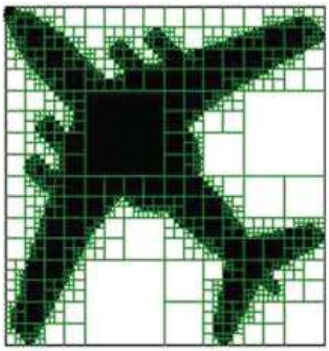


1)

Quad Tree 2)

[문제] $2^k \times 2^k$ 크기의 {0,1} 이미지를 사분 트리(Quad Tree, QT)를 이용하여 인코딩(encoding)할 수 있다. 전체 이미지를 QT를 이용하여 인코딩하는 방법은 다음과 같다. 전체 $2^k \times 2^k$ 이미지를 4개의 $2^{k-1} \times 2^{k-1}$ 사분면으로 다시 나눈다. 이렇게 4개로 나누어진 각 사분면을 다시 재귀적(recursive)으로 하위 단계의 Quad Tree로 인코딩하는 것이다.

Quad Tree는 rooted, ordered tree이며 각 노드의 차수(degree)는 0 또는 4이다. Quad Tree T_Q는 하나의 괄호 안에 표현된다. 해당 괄호 안에는 4개 사분면이 각각 독립된 Quad Tree encoding으로 표현된다. Quad Tree Encoding은 4개의 문자(chars) **{ '0', '1', '(', ')'}**으로 이루어진 문자열이며 이렇게 표현된 것을 Quad Tree String(QTS) 문자열 자료구조라고 한다.

만일 해당 사분면의 픽셀 모두가 0 또는 1이면 해당 사분면 전체는 트리의 leaf node로 표현되면 각각의 label은 **0** 또는 1로 표현된다. 만일 해당 사분면이 0과 1이 섞여 있는 경우에는 다시 4개의 사분면으로 분할한다. 이렇게 분할된 4개의 subtree T1, T2, T3, T4의 Quad Tree의 코드를 각각 **QT1, QT2, QT3, QT4**이라고 할 때 원래 이미지는 각 subtree의 Quad Tree 인코딩 문자열을 다시 괄호로 둘러싼 형식으로 표현된다.

(QT1 QT2 QT3 QT4)

만일 4분면의 모든 픽셀이 '0' 또는 '1'로 구성되어 있으면 더 이상 하위 단계는 표현되지 않는다. 즉 전체가 같은 색의 만일 해당 사분면이 0과 1이 섞인 이미지라면 해당 사분면은 다시 하위 단계의 잘라진 사분면으로 분할되어 recursive하게 encoding된다. 단 이 과정에서 4개의 subtree는 1, 2, 3, 4 사분면 순으로 **왼쪽부터 오른쪽 순서로** 정렬된다. 이렇게 subtree의 순서가 의미를 가지는 가진 트리를 Ordered Tree라고 하므로 Quad Tree는 ordered tree이다.

아래 그림-1에는 2×2, 4×4 크기의 이미지의 Quad Tree, 해당 트리의 인코딩 스트링 QTS

1) 비행기 이미지를 Quad Tree를 이용하여 encoding한 예제.
2) 우리말로는 "사분 트리"로 불린다. 3차원으로 Volume로 확장한 트리는 Oct Tree라고 한다.

가 같이 표현되어 있다.

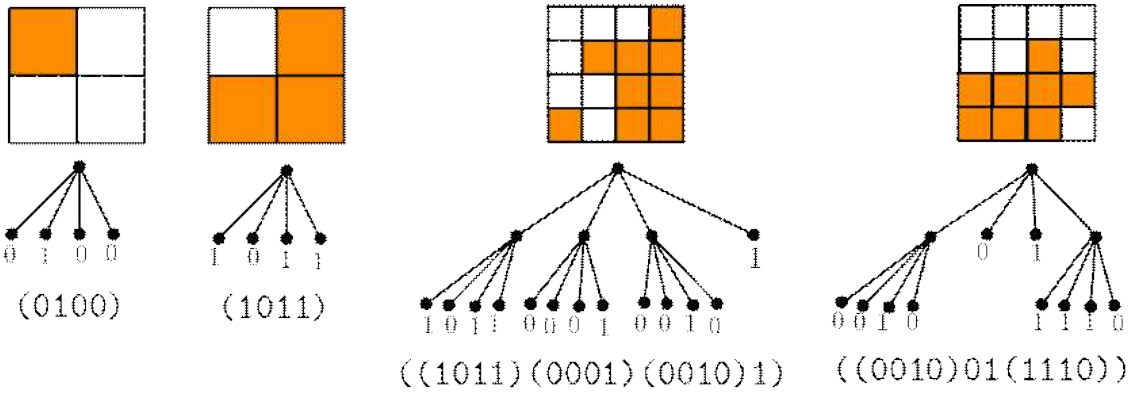


그림-1. 4개의 이미지와 그에 대응하는 QTS 문자열

따라서 만일 흑백 이미지를 전송할 상황에서 이미지 전체의 $2^k \times 2^k = 2^{2k}$개의 pixel을 보내는 것보다 Quad Tree로 인코딩한 QTS 문자열을 보내는 것이 더 효율적일 수 있다.[3]

[입출력] $2^k \times 2^k$ 크기의 이미지에 대응하는 Quad Tree String이 문자열로 주어진다. 여러분은 이 QTS를 읽어서 해당 이미지를 text로 재구성해야 한다. 입력으로 첫 줄에 크기 입력 이미지의 차원 $2^k \times 2^k$를 정의하는 정수 k가 주어진다.

이미지의 차원을 나타내는 정수 k 다음으로는 입력 파일에 있는 데이터가 이미지인지 코드인지를 설명하는 Token이 주어진다. 이 token은 {**IMG, QTS**} 중 하나이다. IMG는 이미지, QTS는 Quad Tree String을 의미한다. 단 이번 과제물에서 k의 범위는 $2 \le k \le 7$ 이다. 따라서 가장 큰 이미지는 128×128이다.[4]

여러분은 입력 파일에서 제공되는 QTS와 IMG에 대응하는 IMG와 QTS를 출력해야 한다. 즉 QTS가 입력이면 대응하는 IMG를, 입력이 IMG이면 대응되는 QTS를 출력해야 한다. IMG의 경우 길이 2^k인 {0,1} 문자열을 2^k개의 줄에 출력해야 한다. 그리고 QTS는 첫 줄에 공백없는 하나의 문자열로 제시해야 한다. 단 이 문자열은 4개의 character '1', '0', '(', ')' 로만 이루어져 있어야 한다. 만일 전체 이미지 모두가 `1`, 또는 0으로 이루어진 image라면 `Quad Code`는 문자인 `'1'` 또는 `'0'`으로 표현된다.

[예제] 아래는 6개의 예제 IMG와 QTS와 대응되는 답을 보여주고 있다. 흥미로운 것은 이미지의 확대 축소를 하더라도 QTS는 변하지 않는다는 것이다. 이것을 Scale Invariant라고 부르는데 이미지 처리에서 매우 유용한 특성이다.

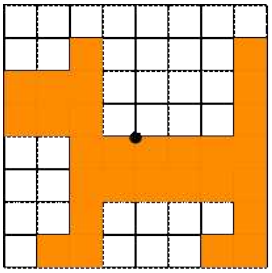
3) 자연적인(natural) 이미지인 경우 QTS로 표현하는 것이 raw image와 비교할 때 더 효율적이다.
4) 과제 이후 실제 사진 크기인 2048×2048으로 실험을 해보길 권한다. 단 흑백 이미지로 바꿔서 해야 한다.

stdin	stdout	stdin	stdout
2 QTS ((1011)(0001)(0010)1)	0001 0111 0011 1011	2 IMG 0001 0111 0011 1011	((1011)(0001)(0010)1)

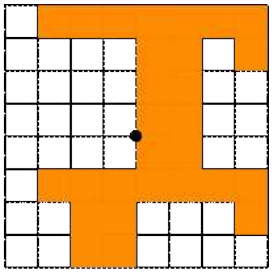
stdin	stdout	stdin	stdout
2 QTS ((0010)01(1110))	0000 0010 1111 1110	2 IMG 0000 0010 1111 1110	((0010)01(1110))
3 QTS ((0010)01(1110))	00000000 00000000 00001100 00001100 11111111 11111111 11111100 11111100	3 IMG 00000000 00000000 00001100 00001100 11111111 11111111 11111100 11111100	((0010)01(1110))

[제한조건] 프로그램의 이름은 **quad.{py,c,cpp}**이다. 최대 제출 횟수는 10회, 데이터 당 제한 시간은 1초, 소스 코드의 최대 token의 수는 700개이다.

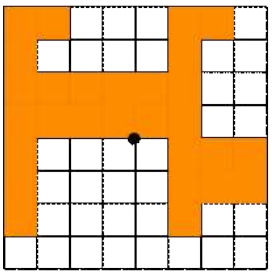
[참고] 아래와 같이 image를 회전하거나 뒤집는(flip) 작업을 pixel 단위로 하지 않고 Quad Tree의 노드 순서를 바꾸는 과정으로 처리할 수 있다. 만일 이것을 $2^{20} \times 2^{20}$ 이미지 space에서 한다면 매우 오래 걸리지만 Quad Tree의 노드 순서만 조정하면 쉽게 처리할 수 있다. 이 뿐만 아니라 이미지의 면적(Number of 1 pixels)이나 둘레(perimeter) 역시도 Quad Tree를 한번 scan하는 것으로 처리할 수 있다. 이미지 처리와 기하작업에 매우 유용한 자료구조이다. 3차원의 경우는 Oct Tree를 활용한다.



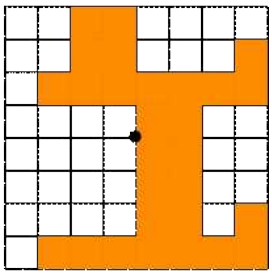
(a) base



(b) Turn 90



(c) Turn 180



(d) Turn 270