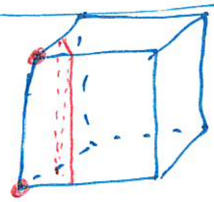
Table filling transforms

hard-code once: 

24 table entries: 12 edges, 2 colors

* how many permutation are possible? (so both have same case)

$$8 \cdot 3 \cdot 2 = \boxed{48}$$

8 ↑ first vertex · 3 ↑ neighbor vertex · 2 ↑ color flip

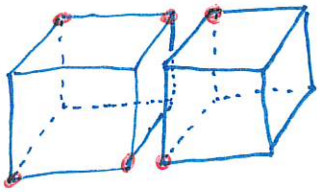
* need a simple, small set of transformations to generate all of them
  - → rotate x axis (or y or z)
  - → rotate $x \to y \to z \to x$ (about diagonal)
  - → flip $x \to -x$ (or y or z)
  - → color inversion
  - → need rot + flip + color

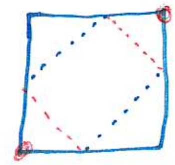
    eg: rot-x rot-y flip-x color-swap
  - → just needs to generate everything, does not need to be efficient
    - → (but rotations preferred)

algorithm

```
fill (case c, triangulation t)
    if c done, return
    fill(c) = t, done(c) = true
    (c1, t1) ← rotx(c,t)
    fill(c1, t1)
    (c2 t2) ← roty(c,t)
    fill(c2, t2)
    (c3, t3) = flipx(c,t)
    fill(c3, t3)
    if not ambiguous
        (c4, t4) = flip color(c,t) ; fill(c4, t4)
```

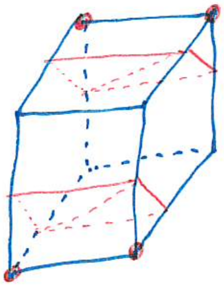
* Marching cubes ambiguity



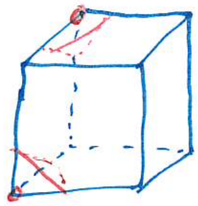


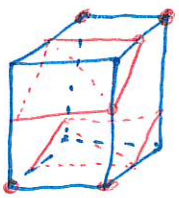
ambiguous face

both cubes must agree • vs •

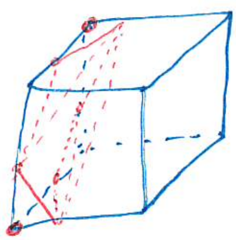


green connected

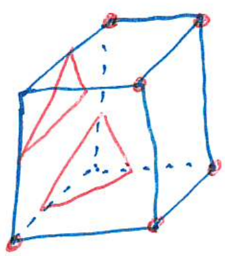




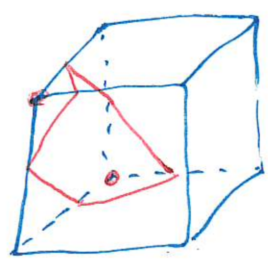
red connected



* Many ways to resolve. eg, always connect the red vertices 

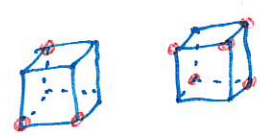


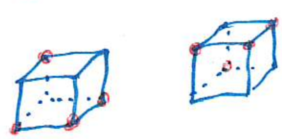
vs



* when filling the table, some transformations do not preserve the resolution

ok	breaks resolution
rotation	color flip
flip	

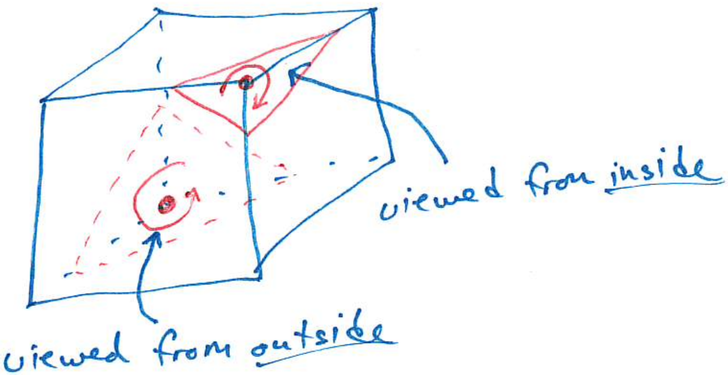
* must create both triangulations for these: 

but not these:  ↕ (flip)

* check for ambiguous face before doing color flip

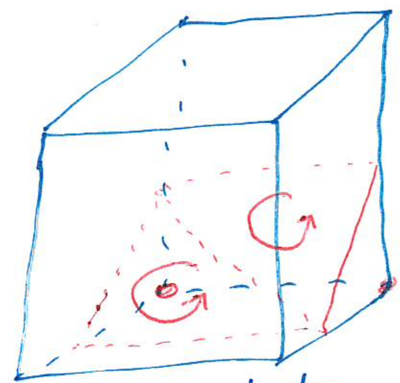
Triangulation and orientation

* CCW viewed from outside (• = inside)



* transformations may reverse this!

→ preserve	flip orientation
rotate	flip
	color flip

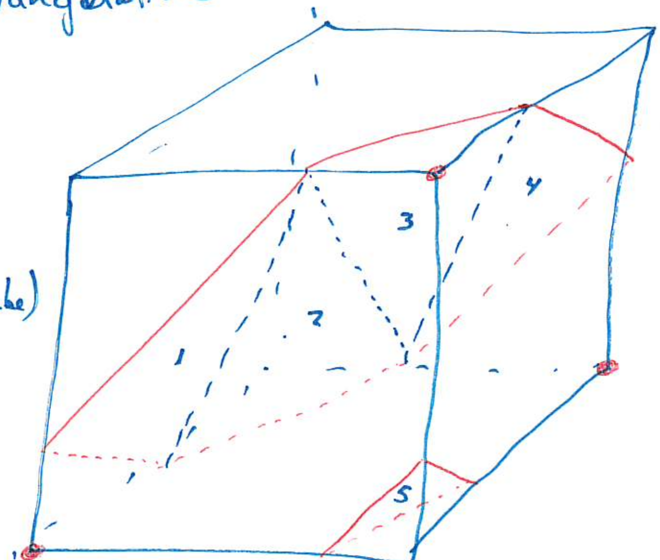


consistent

Other notes

* choose edge numbering wisely - it helps!
* for each xform, make a lookup table mapping (1) old vertex to new vertex &
  (2) old edge to new edge

* use table (1) to compute the new case
* use table (2) to compute the new triangulation
* no tables needed for color flip



* compact representation
  - → 5 triangles (max)
  - → 3 vertices per triangle (each an edge of cube)
  - → 12 edges → 4 bits
  - → $4 \cdot 3 \cdot 5 = 60$ bits
  - → 64-bit long per case encodes everything!
    - → 64-bit = 8 byte · 256 cases = 2KB table (small!)