```
#lang racket
(require racket/trace)


;> (allcombs 2)
;'((0 0) (0 1) (1 0) (1 1))
;> (allcombs 3)
;'((0 0 0) (0 0 1) (0 1 0) (0 1 1) (1 0 0) (1 0 1) (1 1 0) (1 1 1))
(define (allcombs n)
  (cond
    [(<= n 0) '(())]
    [else
     (let ((lst (allcombs (- n 1))))
       (append (map (lambda (x) (cons 0 x)) lst) (map (lambda (x) (cons 1 x)) lst)))]))


;> (cubesort (allcombs 2) 4)
;'((0 0) (0 1) (1 1) (1 0))
;> (cubesort (allcombs 3) 8)
;'((0 0 0) (0 0 1) (0 1 1) (0 1 0) (1 1 0) (1 1 1) (1 0 1) (1 0 0))
(define (cubesort lst len)
  (cond
    [(<= len 2) lst]
    [else
     (let* (
            (newlen (floor (/ len 2)))
            (zerolist (map (lambda (x) (cons 0 x))
                           (cubesort
                            (map (lambda (x) (cdr x)) (take lst newlen))
                            newlen)))
            (onelist (map (lambda (x) (cons 1 x))
                          (cubesort
                           (map (lambda (x) (cdr x)) (reverse (list-tail lst newlen)))
                           newlen))))
       (if
        (= (caar lst) 0)
        (append zerolist onelist)
        (append onelist zerolist)))]))


;> (hamiltonian_cycle_on_cube 2)
;'((0 0) (0 1) (1 1) (1 0))
;> (hamiltonian_cycle_on_cube 3)
;'((0 0 0) (0 0 1) (0 1 1) (0 1 0) (1 1 0) (1 1 1) (1 0 1) (1 0 0))
(define (hamiltonian_cycle_on_cube n)
  (cubesort (allcombs n) (expt 2 n)))


```

```

(trace cubesort)
(hamiltonian_cycle_on_cube 3)
;>(cubesort
;  '((0 0 0) (0 0 1) (0 1 0) (0 1 1) (1 0 0) (1 0 1) (1 1 0) (1 1 1))
;  8)
;> (cubesort '((0 0) (0 1) (1 0) (1 1)) 4)
;> >(cubesort '((0) (1)) 2)
;< <'((0) (1))
;> >(cubesort '((1) (0)) 2)
;< <'((1) (0))
;< '((0 0) (0 1) (1 1) (1 0))
;> (cubesort '((1 1) (1 0) (0 1) (0 0)) 4)
;> >(cubesort '((1) (0)) 2)
;< <'((1) (0))
;> >(cubesort '((0) (1)) 2)
;< <'((0) (1))
;< '((1 0) (1 1) (0 1) (0 0))
;<'((0 0 0) (0 0 1) (0 1 1) (0 1 0) (1 1 0) (1 1 1) (1 0 1) (1 0 0))
;'((0 0 0) (0 0 1) (0 1 1) (0 1 0) (1 1 0) (1 1 1) (1 0 1) (1 0 0))
```