```
#lang racket
(require racket/trace)
; esc control = is uncomment
; esc control ; is comment

;> (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))



;(hamiltonian_cycle_on_cube 10)
;(let ((cycle (hamiltonian_cycle_on_cube 5))) (- (length cycle) (length (remove-duplicates cycle))))
;(hamiltonian_cycle_on_cube 4)
;(hamiltonian_cycle_on_cube 2)
;(hamiltonian_cycle_on_cube 1)
;(hamiltonian_cycle_on_cube 0)
;(hamiltonian_cycle_on_cube -1)



; scrap
;        (append
;         (map (lambda (x) (cons 0 x)) (cubesort (map (lambda (x) (cdr x)) (take lst newlen)) newlen))
;         (map (lambda (x) (cons 1 x)) (cubesort (map (lambda (x) (cdr x)) (reverse (list-tail lst newlen))) newlen))
;         )
;        (append
;         (map (lambda (x) (cons 1 x)) (cubesort (map (lambda (x) (cdr x)) (reverse (list-tail lst newlen))) newlen))
;         (map (lambda (x) (cons 0 x)) (cubesort (map (lambda (x) (cdr x)) (take lst newlen)) newlen))
;         )))]))
;(cons 1 '())
;(cons '((1)) '((0)))
;(append '((1)) '((0)))
;(list-tail '(1 2 3 4) 2)
;(take '(1 2 3 4) 2)
;(expt 2 3)
;(floor (/ 1 2))
;(map (cons 0) '(1 2 3 4))
```