```
#lang racket

(define allcombs (lambda (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)))])))

(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)))]))

(define (hamiltonian_cycle_on_cube n)
  (cubesort (allcombs n) (expt 2 n)))
```