EECS454 Spring 2007-HW4(Solutions)

1. cf. (clrs:Exercises 35.1-3)

The following graph will not yield a ratio bound of two using the proposed heuristic. This gives an example such that the heuristic yields the approximation ratio more than 2(C(H)/C(H*)=13/6>2). The vertices are V = {a1, a2, a3, a4, a5, a6, a7, b1, b2, b3, b4, b5, b6, c1, c2, c3, c4, c5, c6} and the adjacency list representation is given by:

a1**:** b1, b2

a2**:** b3, b4

a3**:** b5, b6

a4**:** b1, b2, b3

a5**:** b4, b5, b6

a6**:** b1, b2, b3, b4

a7**:** b2, b3, b4, b5, b6

Additionally there should be an edge from b1 to c1 and from b2 to c2 and so forth. The heuristic is actually a $\Theta(\log n)$ approximation algorithm. For the upper bound note that the algorithm corresponds to the greedy set cover algorithm, where edges are the elements to be covered and each node *v* represents a set containing the edges incident to this node. By corollary 35.5 we get the upper bound. For the lower bound consider the bipartite graph G = (V ∪ W, E) constructed as follows. Initially, let V = $\{v_1,\ldots,v_n\}$ and W = $\{w_1,\ldots,w_n\}$. First connect v_i to w_i for all *i*. Then add $\lfloor n/2 \rfloor$ nodes to W and connect each of these to exactly two nodes of V not connected to any other of the added nodes. Similarly, continue adding $\lfloor n/3 \rfloor$ nodes and so on. Finally, V contains n nodes and W contains $\Omega(nH_n)$ nodes. Clearly, V is the optimal vertex cover, however the algorithm will (if unlucky) select W. Thus we have shown a lower bound of $\Omega(\log n)$ on the approximation factor.

2.(clrs 34-1:Independent Set)

a.

1) Independent-Set = {<G,k> : graph G has an independent set of size k}. That is, I.S decision problem : Given a graph G = (V,E) and a bound k. Is there a subset V' of V consisting of k nodes such that V' is an independent set?

2) To prove this problem is NP-complete we first show that the independent set problem is in NP. Suppose we are given a graph G=(V,E) and an integer k. The certificate we choose is the independent set V' ⊆ V itself. The verification algorithm affirm that |V'|=k and then for each edge (u,v)∈E it check that at most one of u and v is in V'. This can be done by checking that there is no edge between any pair of

nodes in V'. This is polynomial time since there are $|V'|^2$ pairs to check.

We prove that the independent set problem is NP-hard by showing that clique$\leq_p$IS. This reduction is based on the notion of the "complement" of a graph. Given an undirected graph G=(V,E), we define the complement of G as $\overline{G}$ =(V, $\overline{E}$), where $\overline{E}$ = {(u,v)|(u,v) $\notin$ E}. The reduction algorithm takes as input an instance <G,k> of the clique problem. It computes the complement $\overline{G}$, which is easily obtained in polynomial time. The output of the reduction algorithm is the instance < $\overline{G}$, k> of the IS problem with the same k vertices in the IS as in the clique. The proof that G has a clique of size k iff G' has an independent set of size k'=k (the set of nodes that form a clique in G form an independent set in $\overline{G}$) is very similar to the vertex cover problem proof on p1006 in the text. Using the same examples as given in the book (figure34.15 on p1007) we have G yielding the clique V'={u, v, x, y} and $\overline{G}$ yielding the IS set V'={u, v, x, y}.

b.

for k=|V| down to 1
    if (G has an independent set of size k) // black box: subroutine call
        then return k and halt.
    endif
endfor

Although the independent set decision problem is NP-complete, certain special cases are polynomial time solvable.

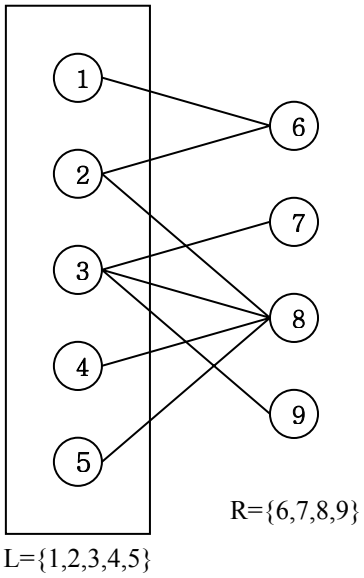
c.

It can be observed that when each vertex in G has degree 2, the graph is just a simple cycle. In this case, an independent set of maximum size can be obtained by starting at any vertex and picking each alternate vertex on the cycle until the size of the independent set is $\lfloor |V|/2 \rfloor$. The running time of this algorithm is obviously O($|V|$)(or O($|E|$) since in this case)

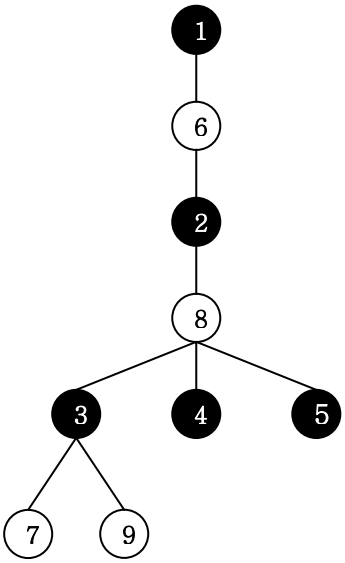
d.

Using figure 26.7 on p665, it can be seen that when graph G is bipartite, then the IS of maximum size is the side with the larger number of vertices (the set of vertices in L). The running time will be O($|V|$).

Example)



BFS tree



A bipartite graph G=(V, E) with vertex partition V =L ∪ R.

L ≡ independent set of maximum size.

Bipartite graph ≡ vertex partition of 2

≡ graph G is 2-coloable.

3.(clrs 35-1:Bin Packing)

a.

We convert the BIN-PACKING problem to a decision problem, as follows:

BIN-PACKING = {<*S*, *j*> | *S* = { s_1, s_2, …, s_n }, 0 < s_i < 1, and all of objects 1, .., *n* fit into *j* unit-sized bins.}

We also define SUBSET-SUM as follows:

SUBSET-SUM = {<*C*, *k*> | *C* = { c_1, c_2, …, c_n }, where c_i is a positive integer, and there is some subset of *C* that adds up to exactly *k*.}

We know that SUBSET-SUM is NP-complete. We now show that BIN-PACKING is NP-hard by showing that SUBSET-SUM ≤p BIN-PACKING. We do this reduction in 2 parts. First, we show that the SUBSET-SUM problem is NP-hard even when restricted to the case where:

$$k = \sum_{c_i \in C} c_i / 2 = W / 2.$$

We will call this instance SUBSET-HALF-SUM. Then we show that BIN-PACKING can solve these instances easily. Now we show that the special instance of SUBSET-SUM ≤p SUBSET-HALF-SUM. Given an instance <*C*, *k*> of SUBSET-SUM, we have 3 cases to consider:

1) If $k = W/2$, then we require no modification.
2) If $k < W/2$, then add an element, *x*, of weight $W - 2k$ to the set *C*. If there is a subset, *A*, that sums to *k* in the old set, then there is a set $A \cup x$ that sums to $k + W - 2k = W - k$ in the new set. Since the new set has total weight $W + W - 2k = 2W - 2k$, this is a problem of SUBSET-HALF-SUM.
3) If $k > W/2$, then add an element, *x*, of weight $2k - W$ to the set *C*. The new set now has total weight $W + 2k - W = 2k$. Then, if a subset in the original set exists that sums to *k*, that same subset exists in the new set, and its weight satisfies $W'/2 = (W + 2k - W)/2 = 2k/2 = k$, which is a problem of SUBSET-HALF-SUM. Note that we are not introducing a false solution, since if a set, *B*, exists in the new problem that does not use *x*, then its complement, *C* - *B*, does not use *x* and has the same sum.

We now show that an instance <*C, k*> of SUBSET-HALF-SUM ≤p BIN-PACKING. First, we convert the elements in *C* to a BIN-PACKING problem of 1-bins to BIN_PACKING by dividing the elements in *C* by *k*. That is, for C = { $c_1, c_2, \ldots, c_n$ }, create $C' = \{ s_i \mid$ such that $s_i = c_i/k\}$. Then, using this modified instance set, *C*', we create an instance <C',2> of BIN-PACKING. If the answer to this BIN-PACKING instance is "yes", then the answer to SUBSET-HALF-SUM is "yes". Otherwise, the answer is "no".

Proof: If the minimal number of bins is 2 and each bin has capacity $W / 2k$, then each much contain a subset of weight exactly $W / 2k$, and each is a solution to SUBSET-HALF-SUM. If the minimal number of bins is greater than 2 (it cannot be 1, since that is smaller than W/k), then no subset with weight $W / 2k$ exists. Otherwise, we would use that subset in the first bin, and the remaining $W/2$ items in the second and have a 2-bin solution.

b.

k unit-sized bins filled to capacity can hold no more than a total weight of *k*. If the sum of the sizes of the objects is *S*, then $S \leq k$. Stated differently, the number of bins *k*, must be greater than or equal to *S*. Since the bins are unit-sized, the minimum (optimal) number of bins needed to hold objects of total size S is $\lceil S \rceil$.

c.

Assume for the purposes of contradiction that the first-fit heuristic leaves two bins that are less than half full. Let b_1 be the first bin that is less than half-full and let b_2 be the second bin that is less than

half-full. Based on the first-fit heuristic, an object, s_k that is in b_2 is placed there because b_2 is the first bin that can accommodate it. But, if b_2 ends up less than half full, then that means that s_k has value < 0.5, which means that it would have fit in b_1, because b_1 has remaining space of more than 0.5. Thus, we have a contradiction with the definition of the first-fit heuristic, in that b_2 is not the first bin that could accommodate s_k. Thus, our assumption must have been false, and the first-fit heuristic must leave at most one bin less than half full.

d.

In the worst case, every bin is half-full (plus some small ε to ensure that 2 bins couldn't be combined). Thus, based on a similar argument to part "b" above, $k/2 \leq \lceil S \rceil$, which implies that $k \leq \lceil 2S \rceil$, given unit-sized bins.

e.

Let C* be the optimal solution, and C be the first-fit heuristic solution. We must show that the ratio of $C/C^* \leq 2$. Based on part "b" above the lower bound on the optimal solution is least $\lceil S \rceil$. Since part "d" shows that the worst case performance of the first-fit heurstic is $\lceil 2S \rceil$, then $C/C^* \leq \lceil 2S \rceil / \lceil S \rceil = 2$, which proves that the first-fit heuristic is a 2-approximation algorithm.

f.

(1) Create first bin (i=1).
(2) Sort objects into increasing size.
(3) Iterate through the objects
    (3a) If object fits in current bin b_i, add that object to bin b_i.
    (3b) If object doesn't fit, create a new bin b_{i+1} and put the object into that bin. $i = i + 1$.
(4) When out of objects, output i as the number of bins needed to hold the objects.

Line (2) to sort n objects is $O(n \log n)$. The iteration in line (3) is encountered $O(n)$ times. The amount of work to perform per iteration is O(1). The rest of the lines are also O(1) work. Thus, the total running time of this algorithm is $O(n \log n)$.