Problem 1

double-map Analysis

The runtime of double-map is an element of $O(n * p)$ where n is the length of the inputted list alod and p is the runtime of the inputted procedure, proc. double-map makes $\frac{n}{2}$ recursive calls. On each call, it makes a number of calls of constant procedures, and a single call of proc, whose runtime I will call p. Since proc is at fastest constant, we can ignore the cost of the constant calls at each step when giving an upper bound. Thus, the total work is $\frac{n}{2}p$ plus the cost of the base case, which is constant and thus can be ignored in an upper bound. The coefficient $\frac{1}{2}$ can then be droped in the upper bound, giving $O(n * p)$.

alt-reverse-helper Analysis

Let A be the runtime of alt-reverse-helper, n the length of alol, and m the length of each of the elements of alol. The only initial call of alt-reverse-helper is on alol with length equal to the length of the list inputted to alt-reverse and with the length of each element equal to one.

$$A(n,m) = \begin{cases} k_0 & \text{if } n = 0 \\ k_1 & \text{if } n = 1 \\ k_2 + DM(n, App(m)) + A(\frac{n}{2}, 2m) & \text{if } n > 1 \end{cases}$$

In the above, DM represents the double-map procedure, and App the append procedure. append's runtime is linear in the length of the first input list, which is m, so it's just an element of $O(m)$. Given the run-time of double-map given above and the definition of big-O, the entire case where $n > 1$ can just be rewritten as $k_2 + nm + A(\frac{n}{2}, 2m)$. Finally, since for the sake of this problem I can assume that n is a power of 2, n will never equal 0, so the recurrence that I must consider is :

$$A(n,m) = \begin{cases} k_1 & \text{if } n = 1 \\ k_2 + nm + A(\frac{n}{2}, 2m) & \text{if } n > 1 \end{cases}$$

To find the closed from of A, I use the following table:

level	I/P Size	cost per node	# of nodes	level cost
1	n, m	$k_2 + nm$	1	$k_2 + nm$
2	$\frac{n}{2}$, $2m$	$k_2 + nm$	1	$k_2 + nm$
3	$\frac{n}{4}$, $4m$	$k_2 + nm$	1	$k_2 + nm$
...	...	...	...	...
$\log_2 n$	2, $\frac{n}{2}m$	$k_2 + nm$	1	$k_2 + nm$
$(\log_2 n) + 1$	1, nm	k_1	1	k_1

The closed form of the A should equal sum of the level costs:

$$\begin{aligned} A(n,m) &\stackrel{?}{=} \sum_{i=1}^{\log_2 n} (k_2 + nm) + k_1 \\ &= (\log_2 n)k_2 + (\log_2 n)nm + k_1 \end{aligned}$$

Claim: For all natural numbers n, $A(n,m) = (\log_2 n)k_2 + (\log_2 n)nm + k_1$

Proof: The proof is by induction on n.

Basis: In the base case, n = 1, m = m

$$\begin{aligned} A(1,m) &= k_1 \\ &= 0 + 0 + k_1 \\ &= (\log_2 1)k_2 + (\log_2 1)(1)m + k_1 \end{aligned}$$

Step: In the inductive step, $n > 1$
Assume the induction hypothesis:

$$\begin{aligned} A(\frac{n}{2}, 2m) &= (\log_2 \frac{n}{2})k_2 + (\log_2 \frac{n}{2})\frac{n}{2}2m + k_1 \\ &= ((\log_2 n) - 1)k_2 + ((\log_2 n) - 1)nm + k_1 \\ &= (\log_2 n)k_2 - k_2 + (\log_2 n)nm + k_1 \end{aligned}$$

Now to show that $A(n,m) = (\log_2 n)k_2 + (\log_2 n)nm + k_1$

$$\begin{aligned} A(n,m) &= k_2 + nm + A(\frac{n}{2}, 2m) \\ &= k_2 + nm + (\log_2 n)k_2 - k_2 + (\log_2 n)nm - nm + k_1 \\ &= (\log_2 n)k_2 + (\log_2 n)nm + k_1 \end{aligned}$$

Q.E.D.

Now to show that $A(n,m) \in O(nm \log n)$

$$
\begin{aligned}
&\lim_{n\to\infty} \frac{A(n,m)}{nm\log_2 n} \\
&= \lim_{n\to\infty} \frac{(\log_2 n)k_2 + (\log_2 n)nm + k_1}{nm\log_2 n} \\
&= \lim_{n\to\infty} \frac{(\log_2 n)k_2}{nm\log_2 n} + \lim_{n\to\infty} \frac{(\log_2 n)nm}{nm\log_2 n} + \lim_{n\to\infty} \frac{k_1}{nm\log_2 n} \\
&= 0 + 1 + 0 \\
&= 1
\end{aligned}
$$

Therefore, $A(n,m) \in O(nm \log_2 n)$. By the logarithm rule, the base of the logarithm can be dropped, thus $A(n,m) \in O(nm \log n)$. Finally, since alt-reverse-helper is defined within alt-reverse and only called in alt-reverse on a list of singleton lists, m can be assumed to always be equal to 1. Since $A(n,1) \in O(n \log n)$ and $m = 1$, $A(n) \in O(n \log n)$.