Бабушкин А.

1 Обязательные задачи

1. dp[i][j] – число разбиений числа i на слагаемые так, чтобы минимальное было не меньше j. dp[i][j] = dp[i][j + 1] + dp[i - j][j].
   Тогда если надо найти k-ое разбиение числа i, то давайте перебирать j, пока префсумма dp[i][j] не превысит k. Когда превысит, мы нашли первое число в разбиении. Уменьшим k, уменьшим i на j и продолжим поиск.

2.

3. 5^n. Каждый бит может быть в одном из пяти положений:
   1) Он есть в маске a, тогда он точно есть в b, но его нет в маске c
   2) Он есть в маске a, тогда он точно есть b, и ещё он есть в c
   3) Его нет в a, и его нет в b
   4) Его нет в a, он есть в b и его нет в c
   5) Его нет в a, он есть в b, и он есть в c

4.
   a) dp[mask][j] – есть ли путь по вершинам из mask с началом в вершине, имеющей наименьший номер среди лежащих в mask и концом в j. Обновление – перебрать все рёбра j -> k и обновить dp. Как именно? Если k имеет больший номер, чем какая-то из вершин mask, то dp[mask | (1 « k)][k], иначе мы её будем считать началом пути, то есть надо обновить dp[mask | (1 « k)][st], где st – прошлая наименьшая вершина, она сейчас станет концом. Если k уже лежит в mask, то ничего обновлять не надо, но если k == st, то надо сказать, что в mask есть гамильтонов цикл.
   b) Заведём новую динамику poss: poss[mask] = other_mask, причём в other_mask лежат те вершины, которые могут быть концами гамильтоновых путей, проходящих по вершинами mask. Теперь научимся пересчитывать poss. Рассмотрим любую вершину k из mask. Если она может быть концом гамильтонова пути, то в poss[mask - (1 « k)] должна лежать хоть одна вершина, имеющая ребро в k. То есть если poss[mask - (1 « k)] & edges_k != 0, то в poss[mask] лежит k. Здесь edges_k – маска вершин, имеющих ребро в k. Но сейчас динамика poss говорит нам не о циклах, а о путях. Чтобы это исправить, давайте возьмём идею предыдущего пункта: poss[mask] содержит множество тех вершин, в которых может заканчиваться путь через вершины mask, начинающийся в наименьшей лежащей в mask вершине. Пересчёт динамики надо тоже изменить так же, как в пункте a.
   Теперь если мы знаем poss[mask], то для каждой вершины в poss[mask] достаточно проверить, есть ли из неё ребро в st, где st – наименьшая вершина в mask. Если есть, то у нас есть цикл по mask. Успех.

5. dp[mask][k] – максимальная стоимость вещей, которую можно унести за 2^k заходов, если вещи брать только из mask. dp[mask][0] считается обычным

рюкзаком, а если k > 0, то можно перебрать подмножества mask и отправить запуститься рекурсивно – одно подмножество M отправить в левую ветку рекурсии, а mask - M – в правую, и обновить dp[mask][k] через dp[M][k - 1] + dp[mask - M][k - 1]. Тогда для каждого k это будет работать за количество подмасок всех масок, то есть 3^n, то, что надо.
Чтобы работало для m, не равных степени двойки, надо сделать именно dp[mask][m] и запускаться от dp[M][m / 2] и dp[mask - M][m - m / 2], работать это всё ещё будет $3^n \cdot \log m$.

6.
a) Переберём все подмножества множеств B за 2^m, поддерживая маску их объединения, для каждого проверим, является ли эта маска покрытием всего множества A, если да, то обновим ответ.
b) Сделаем динамику. dp[mask] – минимальное число множеств из B, необходимое, чтобы покрыть mask. Обновление – перебрать B_i и обновить dp[mask | B_i] = min(dp[mask | B_i], dp[mask] + 1), различных mask у нас 2^n, всё хорошо.

2 Дополнительные задачи

1.
a) dp[n] – число таких перестановок на n элементах. dp[n] = $\sum_{i=k-1}^{n-1} \binom{n-1}{i} \cdot i! \cdot dp[n-i-1]$, т.к. мы выбираем, кто будет в цикле с вершиной n – таких вершин может быть от k - 1 до n - 1, затем i! способами выбираем их перестановку, а затем разбиваем оставшиеся n - i - 1 вершин.
b) Заметим, что $\binom{n-1}{i} \cdot i! = \frac{(n-1)!}{(n-i-1)!}$, а вся сумма, записанная в пункте a), это сумма на префиксе. Тогда давайте вынесем $(n-1)!$ за скобки, получим $(n-1)! \cdot (\sum_{i=k-1}^{n-1} \frac{dp[n-i-1]}{(n-i-1)!} = (n-1)! \cdot (\sum_{j=0}^{n-k} \frac{dp[j]}{j!}$. Тогда сумму можно просто поддерживать, т.к. элементы в ней не зависят от n, а затем домножать на $(n-1)!$ и получать dp[n]. Ура.
2. Пусть dp[n][k] – число разбиений числа n на k слагаемых. Я утверждаю, что dp[n][k] = dp[n - k][k] + dp[n - k][k - 1], если слагаемые различные. Почему? Упорядочим слагаемые по возрастанию. Вычтем из каждого слагаемого единичку. Если первое было единичкой, то оно уничтожилось, и осталось k - 1 слагаемое и число n - k. Если первое != 1, то оно осталось, и у нас всё ещё k слагаемых. У нас не может быть больше одной единички, потому что все слагаемые различные, поэтому других случаев не бывает. чтд.
Осталось только заметить, что если слагаемые различные, то их $O(\sqrt{n})$, поэтому dp будет считать за $n^{\frac{3}{2}}$.

3. Здесь будет написана лажа, Рома, урони её, пожалуйста, а то я сам не справился. Пусть dp[n][mn] – число неизоморфных деревьев на n вершинах, причём минимальное поддерево корня будет иметь размер ≥ mn. Как обновлять?
Один переход очевиден – dp[n][mn] += dp[n][mn + 1].
А ещё есть нетривиальное обновление: у корня есть k поддеревьев размера mn. Давай переберём k. Сколько способов выбрать k деревьев размера mn, чтобы

способы были различны? $\left(\!\binom{cnt[mn]}{k}\!\right)$, где cnt[mn] = dp[mn][0], то есть сколько всего есть деревьев размера mn, а $\left(\!\binom{x}{y}\!\right)$ – количество сочетаний с повторениями из x по y, то есть сколько есть способов выбрать y предметов среди x различных, если мы можем повторяться. Легко заметить, что нас именно это и интересует – сколько вариантов выбрать себе все поддеревья размера mn. То есть надо сделать dp[n][mn] += dp[n - k · mn][mn + 1] · $\left(\!\binom{cnt[mn]}{k}\!\right)$. Осталось научиться считать $\left(\!\binom{cnt[mn]}{k}\!\right)$. $\left(\!\binom{cnt[mn]}{k}\!\right) = \binom{cnt[mn]+k-1}{k}$, то есть мы свели это к обычному сочетанию. Количество сочетаний можно легко подсчитать за линию, но можно и просто поддерживать при увеличении k – ведь $\binom{x}{y+1} = \binom{x}{y} \cdot \frac{x-y}{y+1}$.
давайте теперь посчитаем, сколько это всё работает. Для фиксированных n, mn это работает за $\frac{n}{mn}$, т.к. $k \cdot mn \leq n$. Но тогда для фиксированного n это работает не больше чем $\sum_{mn=1}^{n} \frac{n}{mn} = n \log n$, а для всех n – $n^2 \log n$.