**Практическая работа 3**

**Практическое применение метода динамического программирования на примере реализации игры BlackJack**

**Цель:** «Применить на практике приёмы динамического программирования и реализовать игру “BlackJack” на языке Python 3.\* с различными вариациями правил».

**Содержание**

[Теоретический материал 2](#_Toc24062500)

[1. Общая информация по задаче 2](#_Toc24062501)

[2. Дополнительные правила 5](#_Toc24062502)

[1) Разбиение пар (split) 5](#_Toc24062503)

[2) Удвоение ставки (double down) 5](#_Toc24062504)

[3) Утроение ставки 5](#_Toc24062505)

[4) Отказ от игры, «отступные» (surrender) 5](#_Toc24062506)

[5) Страхование (insurance) 5](#_Toc24062507)

[6) Отказ от карты 6](#_Toc24062508)

[3. Кратко о правилах игры 6](#_Toc24062509)

[4. Примеры партии одного игрока с дилером 6](#_Toc24062510)

[5. Решение задачи максимизации выигрыша в BlackJack (упрощённые правила) 7](#_Toc24062511)

[Задание на практическую работу 11](#_Toc24062512)

[Порядок выполнения первой части задания 11](#_Toc24062513)

[Порядок выполнения второй части задания 12](#_Toc24062514)

[Требования к отчету 13](#_Toc24062515)

[А. Требования к отчету по структуре 13](#_Toc24062516)

[Б. Требования к отчету по оформлению 13](#_Toc24062517)

[Список дополнительных источников 13](#_Toc24062518)

# **Теоретический материал**

## 1. Общая информация по задаче

*Цель игры* – обыграть дилера (крупье), набрав при этом как можно больше очков (максимум очков, не превышающих число 21).

Рассмотрим базовую (американскую) версию игры.

В игре используются колоды карт (от 1 до 8 колод), которые перемешиваются по определенным правилам. Если в игре участвует одна колода, то карты перемешиваются после каждой партии. Если осталась треть всех карт по окончании партии в несколько колод, то карты также перемешиваются. При увеличении числа колод преимущество игрока уменьшается (т.к. уменьшается вероятность получения комбинации «блэк-джек»).

Комбинация «блэк-джек» – ситуация, при которой у игрока набирается 21 очко (туз и одна из карт, которые значат 10 очков). Значения карт представлены в таблице 1.

Таблица 1. Значения карт по очкам в игре BlackJack

|  |  |
| --- | --- |
| Карта | Значение карты |
|  | 10 очков |
|  | 1 или 11 очков (11 пока общая сумма не больше 21, далее 1) |
|  | Значение карты эквивалентно количеству очков (например, 2 – 2 очка, 10 – 10 очков и т.д.) |

В начале игры (до раздачи карт) игрок или игроки делают ставки. После того, как первая карта сдана, к фишкам со ставками прикасаться запрещается. Дилер раздаёт карты (обычно из одной или двух колод карт, но чаще из «шуза», содержащего не менее четырёх колод): по две карты каждому игроку и себе раздаёт две карты, одна из которых открыта, а другая находится на столе рубашкой вверх («hole card»). Все карты открываются сразу (видны и дилеру, и игроку).

Если игрок сразу после раздачи набирает 21 очко (у игрока туз и карта в 10 очков), то в таком случае игроку сразу выплачивается выигрыш 3 к 2 (в 1,5 раза превышающий его ставку). Исключение составляют случаи, когда дилеру первой картой (открытой) попадается 10, картинка или туз. В этом случае существует вероятность, что у дилера также будет «блэк-джек». Если первая карта дилера туз, то игроку с «блэк-джеком» предлагается взять выигрыш 1 к 1, либо дождаться окончания кона игры (и в случае, если у дилера не «блэк-джек», получить выигрыш 3 к 2).

Далее игрокам, у которых не «блэк-джек», предлагается на выбор либо взять ещё карту (в таком случае игрок должен сказать дилеру «карту» или «hit»), либо остаться при тех картах (и той сумме очков), которые у него на руке (состояние «достаточно» или «stay»).

При игре с дилером может возникнуть ситуация «ничья», когда у игрока и у дилера на руках равное число очков (меньше или равных 21). Такая ситуация называется «ровно» или «push» и все остаются при своих ставках, никто не выигрывает и не проигрывает (бывают исключения, когда само казино прописывает правило, что «ровно» не учитывается).

Как правило, если у игрока после взятия новой карты в сумме получается 21, дилер не спрашивает его больше и переходит к следующему игроку.

Если у игрока после взятия новой карты сумма очков превысит 21, то такая ситуация называется «перебор». Дилер произносит «много» и снимает ставку игрока в пользу казино.

В ситуации, когда дилер набирает в начале игры туз или десятку (в некоторых вариациях только при тузе), то сразу смотрится «hole card» (скрытая карта дилера) и если у дилера 21 очко («блэк-джек»), то игра прекращается и все игроки (кроме тех, у кого тоже «блэк-джек»), проигрывают.

Игроки могут «застраховать» свою комбинацию от «блэк-джека» у дилера или в зависимости от карт на руках воспользоваться одной из предложенных дилером возможностей (см. раздел [Дополнительные правила](#_3._Дополнительные_правила)). Нужно учитывать, что шансы получить сразу «блэк-джек» для дилера при открытом тузе равны 4/13, что менее 1/3 доли вероятности в игре. Т.о. базовая оптимальная стратегия не рекомендует брать страховку.

После того, как все игроки завершили брать карты, дилер говорит «себе» и раздаёт карты себе. Общее правило игры BlackJack состоит в том, что дилер должен набрать «себе» не менее 17 очков, даже если у всех игроков на руках карты с меньшим числом очков, чем 17. Обычно это правило написано на игровом столе (если казино ввело другое правило, его также указывают на столе).

Игра каждого из игроков ведется против дилера. Карты и ставки остальных игроков значения не имеют, игра ведется параллельно.

Далее приведем одну из оптимальных стратегий в Таблице 2 (подробнее удвоение ставки, отказ, разбиение, страховку и т.д. см. в разделе [Дополнительные правила](#_3._Дополнительные_правила)). Дилер, как было указано ранее, берет карты до 17 (и выше), одна из карт дилера скрыта. Рассмотрим ряд открытых карт дилера: Туз 10 9 8 7 2 3 4 5 6. Здесь карты расположены по значимости. Туз – самая сильная карта, 6 – самая слабая.

Таблица 2. Базовая оптимальная стратегия поведения в игре BlackJack

|  |  |  |
| --- | --- | --- |
| **Название дейтсвия** | **Количество очков** | **Действие** |
| Ход/пас (твёрдые руки) | 5-8 очков | ход |
| 12 очков | пас, если карта дилера очень слабая (4-6), иначе – ход |
| 13-16 очков | пас, если карта дилера слабая (2-6), иначе ход |
| 17 очков и выше | пас |
| Отказ (твёрдые руки) | 15-16 очков | против очень сильной карты (10 или Туз) |
| Удвоение ставки | 9 очков | удвоить, если карта дилера слабая (3-6), иначе ход |
| 10 очков | ход, если у дилера открыта 10 или Туз, иначе удвоить |
| 11 очков | ход, если у дилера открыт Туз, иначе удвоить |
| Ход/пас (мягкие руки, комбинации с Тузом) | 13-17 (3-7) очков | ход или удвоить против очень слабой открытой карты у дилера (5-6) |
| 18 (8) очков | ход, если открытая карта дилера сильная (9 10 А) или пас |
| 19 (9) очков и выше | пас |
| Разбиение пар | Туз-Туз, 8-8 | разбивать всегда, за исключением открытого у дилера Туза, тогда ход |
| 5-5, 10-10 | не разбивать (при 5-5 лучше удвоить, а 10-10 даёт хорошую сумму в 20 очков) |
| 4-4 | как правило, не разбивать (разбить можно при очень слабой карте у дилера (5-6)) |
| 9-9 | пас, если открытая карта дилера 7 10 или Туз, иначе разбить |
| 2-2 3-3 6-6 7-7 | разбивать при открытой карте дилера 7 и ниже (6 и ниже для 6-6) или ход, если у дилера открыта 8 и выше |

## 2. Дополнительные правила

### Разбиение пар (split)

Происходит, когда у игрока на руке две карты одного достоинства, например, 2-2, 3-3 и т. д., кроме карт с картинками (если они не одинаковые, две картинки считаются как пара). Дилер предлагает игроку split или разбиение одной «руки» на две. Если игрок соглашается, то он вносит на игровой стол ставку, равную предыдущей, дилер разделяет карты на две «руки», раздаёт на каждую по одной карте и продолжает играть с игроком так, как будто он изначально внёс две ставки (игрок играет за двоих).

Исключение составляет split двух тузов: игроку к тузам раздаётся по одной карте, далее он брать карты не может, но может «переразбить» тузы (зачастую разбить тузы разрешено только один раз). Если при раздаче вторых карт опять образуется пара, то дилер предлагает игроку разбить ещё раз (в правилах максимальное число разбиений за игру не указано, но при этом количество разбиений тузов оговаривается отдельно в правилах конкретного казино).

### Удвоение ставки (double down)

Игрок может удвоить ставку и при этом получить ровно одну карту. В некоторых казино такое разрешается лишь тогда, когда у игрока на руке 10 или 11 очков (в некоторых случаях, когда 9 очков). Некоторые казино позволяют игроку удвоить ставку после разделения карт.

### Утроение ставки

Игрок может утроить первоначальную ставку, после того, как сделал удвоение. И получить ещё одну карту. Встречается редко и не является общепринятым.

### Отказ от игры, «отступные» (surrender)

В некоторых правилах BlackJack игроку позволяется совершить «ранний отказ» или снять половину своей ставки после раздачи первых двух карт, если игрок думает, что его набор карт плох (отдельно правилами оговаривается отказ против открытого туза у дилера). «Поздний отказ» бывает при игре по базовым (американским) правилам после того, как дилер посмотрит «скрытую карту» (если открытая карта туз или десятка).

### Страхование (insurance)

Если у дилера первой открытой картой является туз, то дилер может предложить игрокам страхование от «блэк-джека» у дилера. Игрок, в случае согласия, ставит дополнительную страховую ставку, равную половине первоначальной. Если у дилера будет «блэк-джек», то игрок проигрывает свою основную игровую ставку, но выигрывает страховочную. Страховка выплачивается в размере 2 к 1, т. е. игрок возвращает свою ставку полностью. В случае, когда у дилера не «блэк-джек», игрок проигрывает страховочную ставку, но продолжает игру со своей основной ставкой.

Частным случаем страховки является ситуация «равных денег» (еven money). Она возникает, когда у игрока «блэк-джек», а у дилера открытой картой является туз. Дилер предлагает игроку получить немедленный выигрыш 1 к 1 (вместо обычной выплаты 3 к 2 за «блэк-джек»). Принятие такого предложения эквивалентно страховке. Считается, что вероятность «блэк-джека» у дилера близка к 4/13 (отличается из-за наличия части карт на столе), поэтому страховаться или брать равные деньги выгодно только при избытке десяток в колоде.

### Отказ от карты

Игрок уже не может отказаться от карты которую взял.

## 3. Кратко о правилах игры

Каждая карта имеет значение в очках (см. [таблицу 1](#_1._Общая_информация)). Туз считается как 1 или 11 очков, карты с номерами (от 2 до 10) считаются по номинальной стоимости. Лицевые карты (с «картинками») считаются как 10 очков.

Значение «руки» – сумма значений полученных от дилера карт. Кроме того, существует комбинация «блэк-джек», состоящая из туза и любой карты со значением 10 (любой масти 10, J, Q, K),  которая превосходит все остальные «руки» с суммарным значением 21.

Если игрок набрал сумму очков больше, чем 21, то у него «перебор», что означает проигрыш в игре.

Игрок пытается максимизировать свой счет, избегая перебор.

Побеждает в игре «лучшая рука», т.е. наибольшее количество очков «руки», не превышающее 21.

## 4. Примеры партии одного игрока с дилером

Рассмотрим пример для того, чтобы лучше понять правила игры.

Дана колода в 52 карты и дилер раздает две карты игроку и две карты «себе».

У игрока на руке  { Q♦ , 4♣ }, у дилера верхняя карта { 10♣ }

У игрока на руке общее количество очков 14 (или «на руке 14»)

Игрок принимает решение взять еще одну карту и она оказывается { 6♦ }

Итого «на руке» игрока 3 карты  { Q♦ , 4♣ , 6♦ } и общая сумма - 20 очков. Разумно больше не брать (решение - stay), т.к. получен максимальный выигрыш в игре до числа 21.

**Ответить на вопрос.** Какое решение примет игрок в случае, когда разыгрывается всего одна партия и известно, что 1) Туз может считаться как 1 очко при сумме большей, чем 21; 2) существует уверенность, что следующая карта – Туз?

Рассмотрим несколько ситуаций возможного значения «скрытой» карты дилера и его решений, заданных описанной выше детерминированной стратегией:

1. Вторая карта дилера 6♣, «на руке»  { 10♣ , 6♣ } и 16 очков. Согласно стратегии дилера берем ещё одну карту: 3♣ и игрок выигрывает $1 (20 против 19).
2. Вторая карта дилера 7♣, на руке  { 10♣ , 7♣ } и 17 очков. Выигрыш игрока. Если игрок рискнет и возьмет ещё карту, то это ничего не изменит только в случае, когда игрок угадал Туз.
3. Вторая карта дилера J♣ на руке { 10♣ , J♣ } и 20 очков. Ничья.
4. Вторая карта дилера A♥ на руке { 10♣ , A♥ } и 21 очко – «блэк-джек». Выигрывает дилер (проигрыш игрока, если игрок не рискнул) и ставка игрока $1 уходит в «банк» казино.

## 5. Решение задачи максимизации выигрыша в BlackJack (упрощённые правила)

В таблице 3 представлены основные моменты, которые нужно учесть при разработке алгоритма для задачи BlackJack.

Таблица 3. Основные термины, структуры и абстракции в задаче BlackJack

|  |  |
| --- | --- |
| Карта | **Сard - пара {n, s}**  **n** (number, номинал) – кодировка номинала карты, от 2 до 10 для «числовых карт», A (Ace, туз) = 11, J (Jack, валет) = 12, Q (Queen, королева) = 13, K (king, король) = 14  **s**(suit, масть) - числовое значение (может быть перечислением enum в программе), не влияет на количество очков. Пусть масть кодируется числом в интервале [0, 3]. Пример: 0=♠ (Spades, пики), 1=♣ (крести), 2=♦ (буби), 3=♥ (Hearts, черви).  Каждая карта внутри колоды представлена уникальной парой {n, s}:  9♦ => {9,2}, A♠ => {11, 0}, K♣ => {14, 1}, ...  Операция с картами:   1. вычисление очков v({n,s}) = v(n) |
| Колода карт | c[0], c[1], … c[N-1], где N=52 для одной колоды  В программе мы можем представить колоду как класс CardDeck, который содержит:   1. список List<Card>; 2. операцию (функцию) генерации колоды; 3. функцию получения новой случайной карты из колоды |
| «Рука» (Hand) | Карты на руках игрока, выданные дилером с начала игры и/или взятые дополнительно |
| Вычисление очков | Функция вычисления значения/очков v(n)   * v(n) = n, если n <= 10 * v(n) = 1 или 11, если n =11 * v(n) = 10, если n > 11 |
| Валидность игры | Условия, когда игра может продолжаться, например, наличие в колоде необходимого для партии количества карт |
| Партия и понятие игры | Партия – это раздача с учетом ходов игрока и дилера до подсчета количества очков и исхода игры с дальнейшим расчетом вознаграждения каждого участника (может быть -1 проигрыш, 0 ничья, 1 выигрыш).  Играем k партий play(0...k-1), где k – отдельный параметр, который задается в программе. |
| Возможные решения | stay = не брать больше карту, hit = взять карту |
| Ставка | Примем за ставку всегда $1. При старте игрок имеет фиксированное кол-во денег, которое может изменяться только исходами партий в процессе игры. |

Упрощение правил игры для первой версии расчета оптимальной игры состоит в том, что:

1. Игнорируется ситуация split;
2. Значение выигрыша каждой сдачи находится во множестве {-1, 0, 1}, т. е. игрок может выиграть только $1 и игнорируется повышение выигрыша в случае «блэк-джек» у игрока

Ниже на рис. 1 приведен примерный псевдокод функции BJ, которая вычисляет максимальный выигрыш для i сыгранных партий.

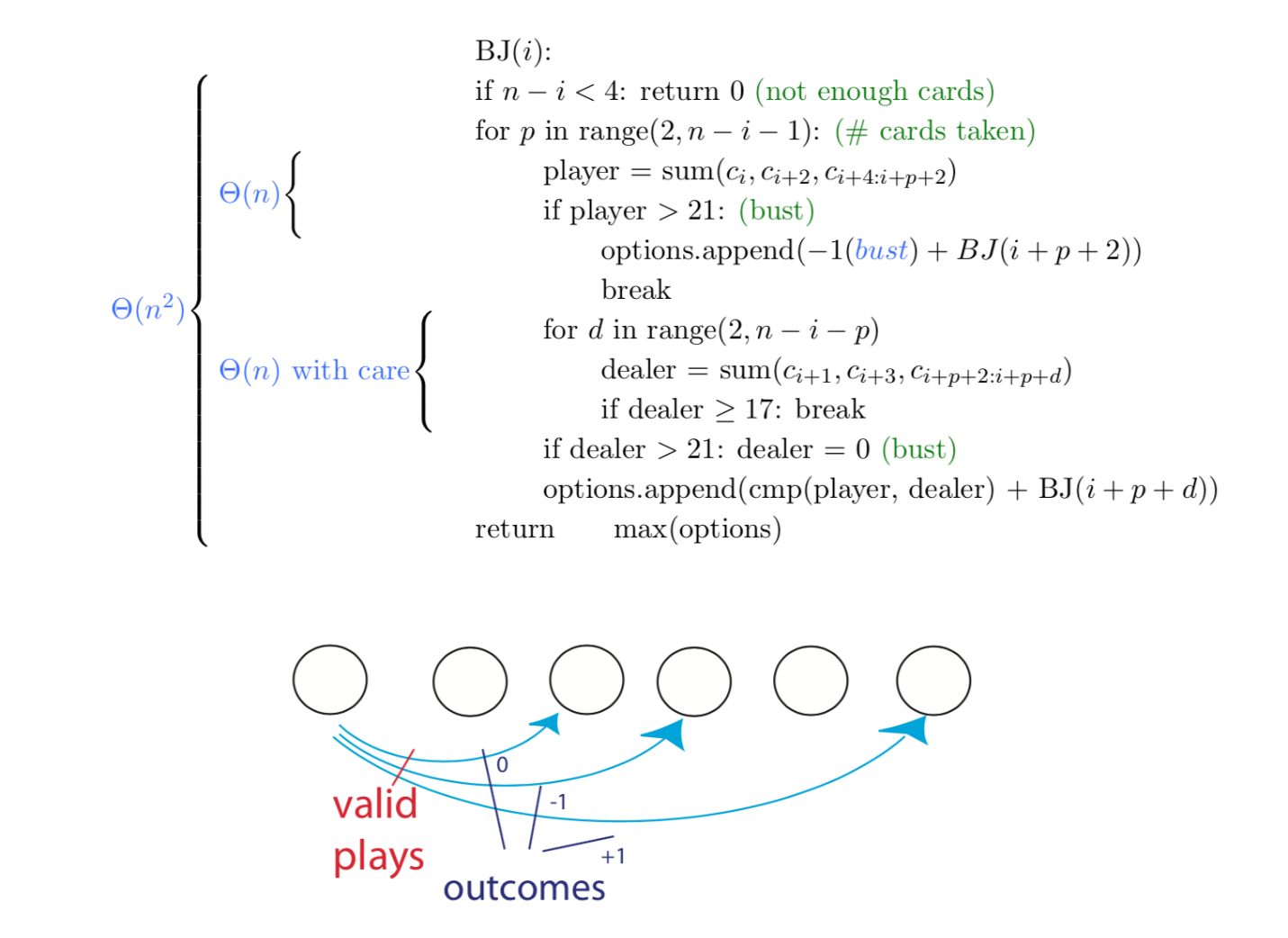


Рис. 1. Вычисление максимального выигрыша для i сыгранных партий

В версии BlackJack, которую необходимо разработать, один игрок играет против дилера (без других игроков). Кроме того, игрок «читерит» и у него есть рентгеновское зрение, которое позволяет игроку видеть всю колоду (c0, c1, ..., cn-1).

Как и в казино, дилер будет использовать известную фиксированную стратегию и брать карты, пока не наберет 17 очков или более (stand-on-17).

Игрок может делать ставку $1 в каждом раунде и может выиграть 1 доллар, проиграть 1 доллар, либо будет «ничья» или 0.

**Ответить на вопросы.** Каким образом можно максимизировать выигрыш игрока в описанной упрощенной версии игры? Когда игрок должен брать еще одну карту (hit), а когда нет (stay)?

При реализации задачи необходимо использовать изученные ранее приемы (или шаги) динамического программирования и найти наилучшую последовательность ходов в игре (максимизация выигрыша).

Предлагается следующая стратегия. Игрок хочет угадать, когда в партии он будет брать карту и когда останавливаться на какой-то последовательности игровых карт, начиная с индекса i.

Пусть i – количество карт, которые уже сыграны, а BJ(i) – лучшая игра с использованием оставшихся карт (). Набор BJ(i) будет являться набором подзадач в игре.

*Сколько всего подзадач?* Поскольку i охватывает от 0 до n, мы имеем *O*(n) подзадач.

*Какими будут «догадки» игрока?* Игрок должен угадать, сколько раз он принимает решение HIT (запрашивает еще одну карту из колоды). Всего *O*(n) вариантов для этой догадки.

Приведем примерную реализацию рекуррентного соотношения, оформленного в виде метода на языке C# (предполагается, что все функции находятся внутри некоторого класса).

// Возвращает исход игры по очкам игрока и дилера (выигрыш для игрока)

public int cmp(int player, int dealer){...}

// Возвращает количество очков по заданным значениям карт

public int GetValue(int number){...}

public int BJ(int i)

{

    if(n-1 < 4) return 0; // т.к. нет больше карт для игры

    foreach(var p in Enumerable.Range(2, n-i-2))

    {

        // Дилер раздает - сначала игроку, потом себе, потом игроку...

        player = GetValue(c[i]) + GetValue(c[i+2]);

        for(var j = 4; j <= i+p+2; j++) player += GetValue(c[i+j]);

        if(player > 21)  // перебор у игрока

        {

            options.Add(-1 + BJ(i+p+2));

            break;

        }

        foreach(var d in Enumerable.Range(2, n-i-p))

        {

            dealer = GetValue(c[i+1]) + GetValue(c[i+3]);

            for(var j = p+2; j <= p+d; j++)

            {

                dealer += GetValue(c[i+j]);

            }

            if(dealer >= 17) break;

        }

        if(dealer > 21) dealer = 0; // перебор у дилера

        options.Add(cmp(player, dealer) + BJ(i+p+d));

    }

    return options.Max();

}

Приведем примерную реализацию рекуррентного соотношения, оформленного в виде метода на языке Python 3.\* (предполагается, что все функции находятся внутри некоторого класса).

# Возвращает исход игры по очкам игрока и дилера (выигрыш для игрока)

def cmp(self, player, dealer): ...

# Возвращает количество очков по заданным значениям карт

def getValue(self, number): ...

def BJ(self, i):

    if(n-1 < 4): return 0 # т.к. нет больше карт для игры

    for p in range(2, n-i-2):

        # Дилер раздает - сначала игроку, потом себе, потом игроку...

        player = getValue(c[i]) + getValue(c[i+2])

        for j in range(4, p+2):

            player += getValue(c[i+j])

        if(player > 21):

            # перебор у игрока

            options.append(-1 + BJ(i+p+2))

            exit(0)

        for d in range(2, n-i-p):

            dealer = getValue(c[i+1]) + getValue(c[i+3])

            for j in range(p+2; p+d):

                dealer += getValue(c[i+j])

            if(dealer >= 17): exit(0)

        if(dealer > 21):

            # перебор у дилера

            dealer = 0

        options.append(cmp(player, dealer) + BJ(i+p+d))

    return max(options)

Можно построить DAG (направленный граф без циклов или дерево решений) для визуализации этой проблемы. Вершины будут представлять подзадачи, ребра будут представлять связи между подзадачами. Весом ребер будет результат (-1, 0 или 1) и они будут указывать на подзадачи с большим значением i (как видно из рекуррентного соотношения).

***Цель для нескольких партий игры BlackJack*** – найти самый длинный путь в дереве решений для максимизации выигрыша в игре, т.е. максимизации суммы исходов отдельных партий внутри игры.

# **Задание на практическую работу**

## Порядок выполнения первой части задания

* 1. Определить структуры данных для игры в программе (классы, перечисления, массивы и т.д.);
  2. Создать консольное приложение для реализации игры. В консольном приложении реализуем ручное управление игрой. Пример интерфейса (возможна другая реализация): “Ваши карты: Q♠ 9♥ (19). Ваше решение: 1 - hit, любая другая клавиша - stay”

В решение входит:

А) Генерация колоды для игры;

Б) Раздача карт игроку и подсчет очков;

В) Моделирование игрового процесса.

Г) Решение задачи оптимизации. Создать проект с тестами (можно посмотреть существующее решение на языке C# и переписать его на язык Python 3.\*). Реализовать функцию для решения подзадач BJ(i) и решить исходную задачу, вычислив BJ(0) на сгенерированной колоде размера N.

## Порядок выполнения второй части задания

1. Создать визуальный интерфейс, описанный в теории, для простой реализации игры BlackJack (первая часть задания) на PyQt5.

Интерфейс игры сделать по своему усмотрению.

Для визуализации колоды в игре необходимо сделать следующее. Создать две папки с картинками для карт. Выбрать оптимальный размер для отображения карт (все картинки должны растягиваться без потерь для выбранного размера и лучше, если карты будут одинковыми или почти одинаковыми).

А) В первой папке разместить картинки для карт лицевой стороной. Название картинкам сделать с учетом числовой кодировки их масти (0=♠, 1=♣, 2=♦, 3=♥). Например:

cards/...jpg

2\_0.jpg // 2♠

2\_1.jpg // 2♣

2\_2.jpg // 2♦

2\_3.jpg // 2♥

3\_0.jpg // 3♠

…

10\_3.jpg // 10♥

(далее для краткости x - это масть)

11\_x.jpg // A = Ace (туз)

12\_x.jpg // J = Jack (валет)

13\_x.jpg // Q = Queen (королева)

14\_x.jpg // K = King (король)

Б) Вторая папка должна содержать отдельно “рубашки” карт (того же оптимального размера, что и карты) – минимум две разные для смены колоды в программе.

Например:

cover/...jpg

1.jpg // изображение рубашки того же размера, что и карта

2.jpg // альтернативная рубашка

1. \* Добавить в программную реализацию игры расширение (по заданию преподавателя и после реализации первой части задания и пункта 1).

# **Требования к отчету**

## А. Требования к отчету по структуре

1. Титульный лист;
2. Постановка задачи;
3. Алгоритм решения;
4. Программа;
5. Скриншоты выполнения;
6. Общий вывод. Ответить на вопросы в тексте практической работы. Сформулировать почему задача является задачей динамического программирования.

## Б. Требования к отчету по оформлению

1. Шрифт отчета Times New Roman (шрифт кода программ может отличаться);
2. Нумерация страниц для всех, кроме титульной.

# **Список дополнительных источников**

1. Пример реализации кода игры (на C#) можно посмотреть по следующим ссылкам на github:

1) <https://github.com/jonny-novikov/BlackJack>

2) <https://github.com/Egorka99/BlackJack>

3) <https://github.com/veldm/BlackJack>

1. Статьи на Википедии ([Русский](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D1%8D%D0%BA%D0%B4%D0%B6%D0%B5%D0%BA), [English](https://en.wikipedia.org/wiki/Blackjack)).
2. Ссылка на лекцию по ДП и игре BlackJack на английском: [MIT Course:  Introduction to Algorithms (2011)](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/recitation-videos/MIT6_006F11_rec20.pdf)